1999

Java Challenge Software Project

Karuna Annavajjala
West Virginia University

JAVA CHALLENGE SOFTWARE PROJECT

Karuna Annavajjala

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Computer Science

John R. Callahan, Ph.D., Chair
James D. Mooney, Ph.D.
Bojan Cukic, Ph.D.

Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
1999

Keywords: Java, Security, Automation, Programming Contests, Software Development

ABSTRACT

Java Challenge Software Project

Karuna Annavajjala

Programming contests are a means of exploiting the problem solving capabilities of developers and they provide a forum for display of extraordinary programming skills. The Java Challenge (JC) Software Project is the saga of creating an automated, secure and responsive programming contest system for deployment on the Internet and to collect information about programming practices, habits, and trends in coding in such restricted environment. The methodology followed to design, implement, and evaluate such a system uses new technologies such as the WWW, mail filtering and sandboxing techniques. The current Java Challenge implementation runs the Java Challenge on a Solaris 2.6 platform under specified regulations. The scripts are developed in Perl. The security features of jdk1.2 have been researched and successfully implemented. The mode of entry acceptance is electronic mail in a specified format. Standard Unix features have been used for data archiving and information redirection. The JC software is an application package that conducts programming contests in an automated manner, provides a secure environment for evaluation and does web listing updates automatically.

Acknowledgements

My sincere gratitude goes to my advisor, Dr John R Callahan whose inspiration and efforts have gone a long way in the successful completion of this thesis. I thank him for his guidance and help.

I wish to thank my parents and my sister for their unconditional love and support. Their trust in my decisions guides me in the right direction. Also, I wish to recognize the efforts of all my teachers who have helped shape my life.

I wish to thank my friends Reshma, Harjinder, Raji and Ron for their encouragement and companionship. I thank Smitha Jammi for her guidance during my college days.

I thank Dr. James Mooney and Dr. Bojan Cukic for their advice and for serving on my committee.

I thank David Krovich, Jarod King and Nick Benders from the Systems group for their timely suggestions and ever-willingness to help. The financial assistance provided by the CSEE department in the form of Graduate Teaching Assistantship is gratefully acknowledged.

Many thanks are due to my friend and soulmate Clarence Gadapati, whose love is my inspiration, whose trust is my motivation and who gives meaning to my life.

To his tremendous love and inspiration, I dedicate this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In any science, by the rule of thumb for principles, put it to a test and you shall know its worth. Computer science is no different in most senses. The concepts and principles of computer science are so applicable to real life situations that there is an ever-increasing need for improvement and better efficiency. Programming or developing code is a major part of the concepts and applications of computer science. Coding is the process of creating software packages using the tools provided by computer technology to satisfy the processing needs. Skillful programming is the ability to produce code that performs its function correctly, consistently and optimally.

The various aspects of programming that are considered for judging its quality include design modularity, ingenuity, reusability, portability and efficiency. Though a quantitative analysis of these aspects is not possible, as they are highly subjective and relative to the context under consideration, a fair classification can be provided by means of comparative studies. This leads to the subject of organizing programming challenges to test the mettle of programmers and study the trends of coding practices in general. This exploits the programmers' skills and displays the special capabilities of the language or tool chosen to accomplish the difficult task. A programming contest is a popular means of judging programming skills and reviving fast track coding capabilities.

A programming contest entry is different from a software project in many ways. The goal of the contest entry is to fulfill the challenge requirements in the minimum possible time in an optimal manner. The software project on the other hand is not so time

sensitive. It involves laborious phases that ensure that the end product not only performs the task at hand but also satisfies the higher principles of software engineering. A programming contest entry, though not always very reusable or portable, must definitely fulfill the required objective in probably the shortest time possible and in the most efficient manner.

A programming contest poses challenging questions that boggle the minds of people having tremendous experience of coding and thus keeps them active and alert to the current state of the art concepts and developments. This is also a channel for promoting advanced study of superior algorithms for various classes of problems. The challenge demands the study and creation of optimal algorithms for finding a solution to the posed problem.

Participating in a contest of this kind is one thing and enabling the running of such contests is an entirely different issue. Development of a package for conducting such programming contests is a full-fledged software project that involves all the phases of a software life cycle. The main concerns while running a contest are validity of entries, automation of the entire process, security of the system and notification and exchange of information among the contestants.

The increasing popularity of programming contests, and the involvement of serious professionals in organizing such events, provided the motivation for the JC software project. It attempts to provide a complete and secure environment and system for conducting programming contests through e-mail. Suitable efforts have been put in to ensure the safety and efficiency of the system. There are provisions for optional

automation of processing and the automatic update of the score board maintained on the server's web page.

Keeping all issues in mind, the specifications of the entire project were drawn up to formalize the requirements and scope of the package. Design decisions involved the arduous task of studying various options available, evaluating their ability to fulfill the requirements in a meaningful manner and then making the most appropriate choices. The implementation was then carried out to form the entire package.

The Java Challenge Software Project is an attempt to create a complete, independent environment for conducting programming contests. All the requirements have been studied and meticulously handled for superior performance. The current version is a fully functional package with scope for further improvement.

A lot of study and research has gone into the making of this software and its successful usage for appropriate purposes. Studies were conducted to learn about similar tasks undertaken by different informal groups and formal organizations. This provided an insight into the issues to be dealt with, the limitations of the available systems and complexity of the entire project.

The JC Software Project has created the system for running the Java Challenge and it now provides a platform for conducting software-engineering studies on various aspect of programming practices. The entries serve as the input to the contest. The feedback and information collected from the contestants participating in the contest provide the real time data that is being analyzed in an attempt to classify coding practices. The data is used to study the error rate of software and conduct comparative studies on

the trends in programming under restrictive conditions, relative performance issues and efficiency of code.

This thesis is a compilation of all the work involved in the JC software project development and the final functional application as it exists today. It describes the various phases in the package development, how various issues were tackled and the ways in which it is useful as an aid for software engineering research. The tasks attempted towards meeting the above objective are explained in detail in the following chapters.

Chapter 3 is a compilation of all the research work in related area and literature survey undertaken before the commencement of the project. The design considerations are discussed in Chapter 4. Chapter 5 describes the actual implementation details of the whole Java Challenge software project. Chapter 6 discusses the various testing methodologies followed to ensure that the software created is robust and of good quality. Chapter 7 has the implementation details of the data collection part of the project. In Chapter 8, the results and conclusions drawn from the entire endeavor are presented. Finally, Chapter 9 proposes some future recommendations and directions for enhancement of the package for more extensive usage.

CHAPTER 2

OBJECTIVES

2.1 Problem and Solution

The problem under consideration and research was to create an automated, secure and responsive programming contest system for deployment on the Internet and to collect information about programming practices, habits and trends in such a system. The solution to this chosen problem has been designed by the design, implementation and evaluation of such a system with the use of new technologies like the WWW, mail filtering and sandboxing techniques.

2.2 Objectives of This Research Work

The objectives of this research are:

- Creating a software package for conducting the programming contest, Java Challenge, via e-mail facility
- Researching similar attempts and work in related fields undertaken by other developers
- Study and analysis of various options available for such a project
- Developing a fully automated system of evaluation and result posting
- Ensuring the safety and protecting the security of the system on which the contest is being conducted
- Trapping and rejecting invalid and illegal code entries to the contest by implementation and use of appropriate handlers

- Study of various software engineering analysis work undertaken in related areas
- Collection of data regarding the efforts of contestants and programming practices
- Inferring results and drawing conclusions from the analysis of data in the context of coding practices in a constrained environment

CHAPTER 3

RELATED WORK

3.1 Introduction

Programming contests have been held in information technology organizations for a long time to promote friendly competition and develop inspiring code. On a smaller scale, such contests are held regularly as a source of intelligent entertainment for the software members of the team. Most contests held in this manner encourage group entries. The members of the group work together on the posed challenge and work out a feasible solution. There are monetary and recognition awards for the winners in such contests. Events of this kind create an active and invigorating work environment.

3.1.1 POTM Contest

A branch of AT&T conducted the Programmer of the Month (POTM) contest for its employees, which later become open to the general public [3]. The POTM contest poses a challenging programming question on the web once a quarter and the interested parties respond to it through e-mail submissions. Results and all other relevant information are maintained on the web site. There are no material awards and the contest is purely for coding fun.

This contest is conducted on a Sun4u Sparc SUNW, Ultra-2. The languages permitted are Perl, Java, C, C++, Fortran and shell scripts. There is a limitation on the size of source code and size of executable file created for the entry to be considered valid. The entries are judged and the best ones chosen based on the execution time of the

program entry. This entire process is mostly manual with the use of simple scripts for some regular repetitive tasks.

With the emergence and spread of web technology, programming contests have gained a lot of attention and popularity. Contests can now be organized and conducted via the World Wide Web without the need for the contestants to be physically present at a location for participation in the event. This expands the scope of the contest, gives more people a chance to participate and respond and also promotes good international understanding in the technical arena as the contestants are from all over the globe.

3.1.2 IOCCC

The International Obfuscated C Code Contest (IOCCC) is one of its kind that professes and patronizes the most obscure C programs [4]. Though seemingly strange and weird at times, this contest is of great significance as it displays the importance of programming style in an ironic way. It stresses the compilers with unusual code and illustrates fine subtleties of C language. That this contest is considered a serious affair can be judged by the fact that the winners for the 1998 IOCCC were announced at IOCCC BOF during the Usenix'99 Technical Conference.

The IOCCC is conducted via e-mail with the contestants sending in entries in a specified format by e-mail. The contest has strict rules about the length of code, the headers and the per-processor directives. To emphasize the importance of interpretation, a separate section is dedicated to the entries that make the worst abuse of the rules of the contest! The evaluation of the entries is done on an individual basis manually on a POSIX compliant platform.

The e-mail entry content is unpacked into files to separate the header message, the author's information, remarks, the build section and the main code. The text sections are scanned to retrieve necessary information for running the C program. Then the code is passed through the C pre-processor for beautification and clean up. This code is then linted, compiled, executed and tested to determine its level of obscurity.

3.1.3 CSCC

Another issue of interest is in the area of promoting computing science knowledge and inspiring more young minds to take to modern tools of technology by spread contests among high school children. The base of programming is an algorithm, and the foundation of any algorithm is logical and analytical thinking. High school is the right place to nurture these skills in the minds of students, as they are open to new ideas, more receptive to friendly challenges and ever so keen to learn and build upon what they already know.

The Calgary Schools' Computer Competitions (CSCC) organized and maintained by The Department of Electrical and Computer Engineering, University of Calgary, Canada, is one such event [5]. The CSCC is a collection of varied kinds of competitions in computer related fields. There are separate sections that deal with each kind of specialty such as Computer based art, web-based programming, creative programming and hardware simulations.

This competition provides a platform for various schools in that region to display their talent. Being free of any system restrictions, the contestants have the opportunity to explore the capabilities of various computing tools and use those to their advantage. This

is not simply a coding exercise, but there are provisions to emphasize the importance of documentation, originality, presentation, enhancement capability and robustness. These are the vital ingredients of any software undertaking and it is thus important to emphasize their role to programming novices who will carry the experience and knowledge a long way into the future. This contest being so varied and broad based, does not use a standard code for evaluating and grading the entries. The entries are judged by different groups of judges each considering a particular aspect of the task performed.

3.1.4 IOI

The International Olympiad in Informatics (IOI) is an International contest in the discipline of computer science for senior students in secondary schools all over the world. IOI is an annual contest and is held at different places in various countries every year. This idea was proposed to the United Nations Educational, Scientific and Cultural Organization (UNESCO) by the Bulgarian delegate Professor Sandov [6]. The IOI is one of the five Olympiads held in various branches of Science. This event not only provides a forum for a variety of programming issues but also attempts to popularize good publications in the appropriate fields by recommending study material for the tasks assigned.

This is a venue based contest unlike others that are web based or e-mail entry based. The participants gather at the venue on the specified date and work on the tasks for two competition days on personal computers. The committee then judges the results. Recreational and cultural events are organized during those days to promote international tolerance and amity.

3.1.5 USACO

The USA Computing Olympiad (USACO) is the selection ground for the participants from USA in the IOI [7]. The USACO conducts Internet hosted contests in various fields of computer science with a wide variety of entry criteria and problem specifications. The contests emphasize individual entries as it increases the level of programming expertise and enables an improved selection of outstanding performance. The chosen candidates are invited to attend the training camp to prepare for contesting in the IOI. The USACO also conducts a National Championship, which is a proctored competition held for five hours at local schools. There are other contests like the Challenge of the Week maintained via the e-mail facility.

3.1.6 ASCL

Another competition is held by the American Computer Science League (ACSL) specific to the schools all over the United States of America [8]. This is a distributed contest held at all participating schools at the same time. It is open to all grades and the contestants are 3 or 5 person teams. The competition is conducted and managed by individual schools with the questions, input and output data being provided by ACSL. The faculty advisors at each school grade the performance of the contesting teams and then the consolidated results are sent to ACSL for tabulation.

There is no specific software used for this entire process as ACSL simply receives the output data from various participating schools and decides if it matches their solution or not. As an expression of motivation and encouragement, the outstanding entries' creators are given prizes of good value like computer peripherals and useful books.

All the contests that are held at school level are designed keeping in mind the low-level problem solving capacity, the restricted analytical ability and limited knowledge base of secondary school pupils. A more practical and meaningful contest would be to design contest tasks based on real-life problems. Such problems would exploit the computing capabilities to the maximum and render excellent services to studies and improvement in problem solving techniques.

3.1.7 ACM Collegiate Programming Contest

Students at college level are more exposed to computing techniques and have a better understanding of the entire software scenario. They constitute the population in the institutes of higher education that is at the threshold of building the workforce of the commercial world. The facilities and technical support available at most colleges is the latest technology that serves well as the platform for honing the programming skills of the interested students. In view of the education system that provides exposure to detailed principles of algorithms and programming language concepts for students studying introductory and advanced computing and mathematical techniques, contests can be more specific, restrictive and yet innovative and challenging enough to attract creative minds.

The Association for Computing Machinery (ACM) is a prestigious international institute dedicated to the promotion and advancement of science and information technology application among students and computer professionals. As a part of the agenda for fulfilling its mission of providing a standard international podium for computing skills, the ACM organizes the International Collegiate Programming Contest

on an annual basis [9]. The contest is preceded by a series of regional competitions that form the teams to participate in the world finals.

For the regional ACM contests, all continents are divided into an appropriate number of regions. Teams are sent from the institutions depending on their regional affiliation. The contest is held for a fixed duration of time with the evaluation based on number of problems solved and the total time taken by the team. Every team works on a single workstation. The languages allowed at the regional level are C and C++ with the addition of Pascal and Java for the World finals. The entries in these contests are judged purely based on the output for the test-input data, which is not revealed to the contestants. This kind of competition is a test of understanding, interpretation, problem analysis and solving and programming skills.

3.2 Evaluation Methodology

Programming contests are an efficient method of inspiring superior computing capabilities and provide a means for judging the skills of the contestant in various aspects of programming. For any contest to be successful in its goals, it needs to be conducted in a consistent and fair manner. There are various ways of evaluating the contest entries and different methodologies for conducting the contests that mainly depend on the contest requirements and specifications.

3.2.1 Manual Evaluation

The most common and primitive method of conducting a contest and evaluating the entries is to do the entire process manually; possibly using simple shell scripts for recurring tasks. This involves posting the questions, accepting the response via e-mail or forms on the web, evaluating the entries by going through the code to ensure legitimacy, compiling and executing it and then comparing the output to judge if the entry performs the task correctly.

An advantage of this kind of set up is that the contest problems need not be very restrictive and definite. The manual judging process provides a scope for the contest to be held in highly interactive fields involving applications and tools that have numerous graphic or multi media capabilities. Thus the contest serves not merely as a testing ground for problem solving and programming skills but also enables testing of analytical and interfacing skills.

Though seemingly simple and straightforward, this method has certain disadvantages and loopholes. The entries generally stream in at a constant rate all through the contest period, which requires that the judging panel always be available to go through the entire evaluation procedure. There are security issues to be taken care of to ensure that the contest entries do not pose a threat to the security of the system running the contest. There have been instances of studies undertaken to analyze the impact of unsafe code on the security of the system, especially in the context of running a programming contest [10]. Such a contest cannot be very broadly conducted due to security and accessibility limitations.

For contest settings, software has been developed that assists the conducting of contests in numerous ways. Such a package would typically handle various possible settings and contain provisions for various modes of entry acceptance, evaluation and result notification.

3.2.2 On-Site Evaluation

Contests that are held at particular sites for limited time periods can be handled quite successfully by manual evaluation due to the limited duration and fixed number of entries. The system specifications of the contestants' entries make the evaluation process simpler and easier to accomplish by on-site evaluation. There are no major system security issues involved in such environments. There is more reliability as far as the contestants' intentions are concerned due to their continued presence and answerability to the committee conducting the event. The system can be temporarily set up to accommodate the challenge requirements and include provisions for the contestant's technical requirements.

3.2.3 Web and E-mail Based Contests

With the emergence of web technology, programming contests are now increasingly held on the World Wide Web or via the electronic mail facility. The tasks assigned in these contests are more restrictive and well defined. They deal with very specific problem descriptions and are usually judged on a comparative basis. The problems are more algorithm design and problem solving skill oriented. It is not the number of features implemented that matter but the efficiency of the chosen design that

has been incorporated in the implementation that is the deciding factor. These contests give rise to different security and implementation concerns. There are major security issues to be handled to ensure that the entries do not perform any malicious actions. The contest entries need to be carefully studied before being allowed to enter the system setup for execution.

Due to the widespread availability of Internet based challenges, the evaluation technique needs to be systematically automated for faster and efficient processing. Web based contests provide an opportunity for a vast range of individuals to participate. This, in effect, increases the probability of intentional or experimental malice. The interface to accept entries via the web needs to be secure and robust. Also, there must be an appropriate method of authenticating the true author of the entry sent in cases where material rewards are involved. A programming contest conducted on the web has wider participation and is more popular as it appeals to all classes and levels of people without any strict restrictions on educational qualifications and age.

The contests run via e-mail have a slightly different flavor. They are widely spread in their response, like the web-based contests, but tend to have fewer uncertainties regarding the contestants' credentials. The entry sent in is an address in itself. It gives information about the source of the e-mail, which can be traced if required. The problems posed in a contest held via e-mail have to be concise and specific to provide for the functioning of e-mail filters and conversion techniques. The electronic mailing system limits the problems to be solvable with a single comprehensive algorithm. There cannot be multiple files or multiple formats as it is expected that the entry be sent as the content of the mail in its entirety. This method of conducting the programming contest depends

on the host system setting for the evaluation and automation procedures. The contest is specific to particular machine architecture and all entries need to execute successfully on that system to be considered valid entries. Proper provisions must be made for archiving the contest responses.

3.3 Evaluation Software

There are different methods followed for conducting programming contest via the web or e-mail. As stated earlier, the manual processing is not an effective solution for these scenarios. There is software developed by different institutions for this kind of work.

3.3.1 NETJUDGE Software

The NETJUDGE © software is one such complete package. Developed by the Universidad De Valladolid, it provides the environment for running an online real time programming contest via the e-mail system [12]. This software prohibits the use of any file input/output and system calls of any kind. The contest is conducted in C, C++ and Pascal languages. The entries are accepted in a prescribed format through e-mail. The format of the body of the e-mail message incorporates the user identification number, problem and solution description.

The Online Judge Software attempts to evaluate the entry by compiling executing and tabulating the results. There is a provision for automated e-mail notifications regarding the various stages of processing. There are special traps and exception handlers for cases of entries sent from non-authenticated sources or from Internet address to which

access is denied by the system. There are restrictions on size of source code and the time and memory limits for executing the entry on the system.

Netjudge also creates and maintains logs for detection of inappropriate usage or attempt of hacking. This project is still under development as enhancements are being designed for implementation on a Linux system and to create a submission tool that will overcome the size limitations.

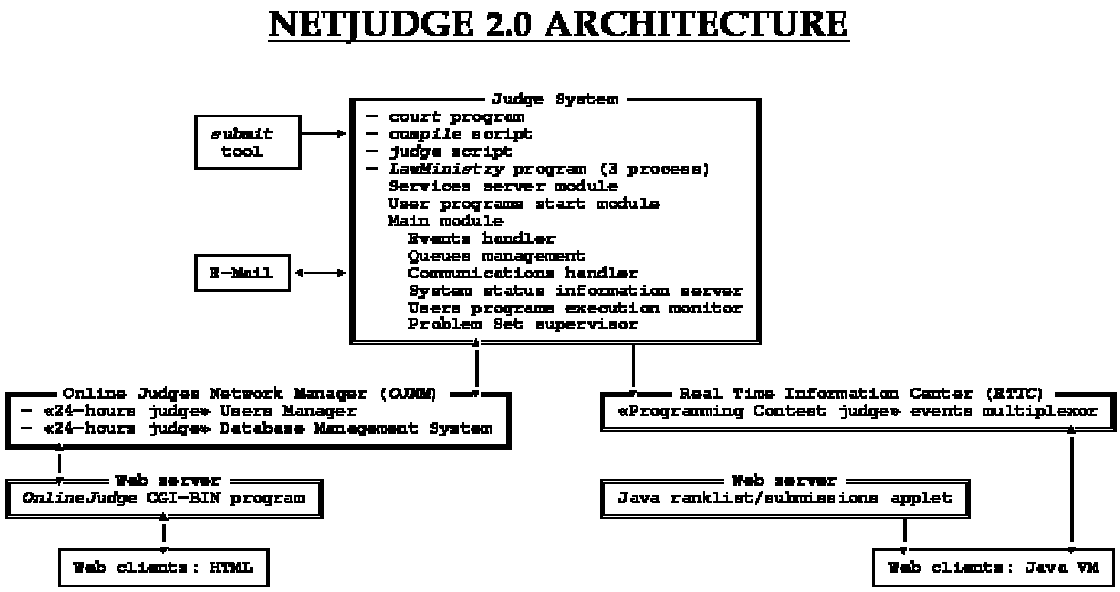

Figure 3.1: Architecture Setup of NETJUDGE Software [12]

The software implementation is done on various levels of evaluation and there are separate modules for different kinds of functions. The design is based on queue management processes and scripts for various operations. The scheme incorporates servers for rank listing and for common gateway interface programs.

The internal scheme shown in figure 3.2 depicts the way the entire system works. The processing is parallel and there is true automation and immediate updating of rank listing by the software itself.

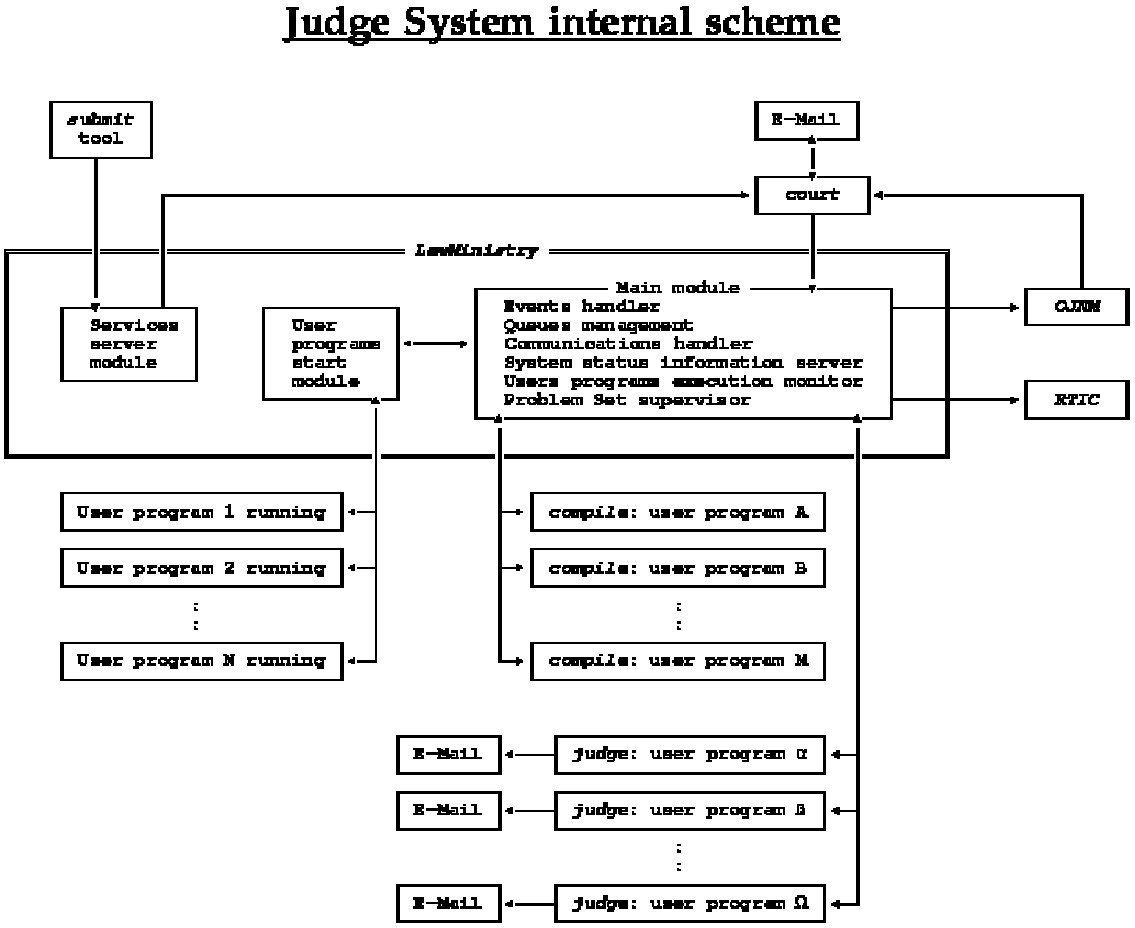

Figure 3.2: NETJUDGE Internal Scheme Diagram [12]

3.3.2 PCCS

Another package that satisfies the requirements of running a programming contest in different computing environments in limited site locations is the Programming Contest Control System (PC^2). The P-C-squared is software developed by the students of California State University, Sacramento as an aid for evaluation and result posting during programming contest [11]. The latest version of this software written in Java was used at the International Olympiad of Informatics at Eindhoven, Netherlands.

P-C-squared package has a wide variety of configurable options for specifying the contest rules, the language involved, the scoring methodology and result notification and score update frequency. This package can be used only in contests where the entries are accepted on floppy disk or on the network. The earlier versions supported only LAN submissions but the software was later enhanced to include floppy disk submissions.

3.4 Software Engineering Studies

As mentioned earlier, programming contests serve as a means of conducting software engineering related studies of programming practices. Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines [13].

This view is as appropriate now as it was years ago. In view of the various facets of software engineering, it can be stated that software engineering is not same as programming, even though programming is an important ingredient of software engineering.

One definition that provides the crux of software engineering states, Software Engineering is the systematic approach to the development, operation, maintenance and retirement of software [14].

Software engineering studies range from testing the software for correct functionality to assessment of its reliability and verification and validation of the design with regards to requirements and specifications. Reliability of software is the probability of its failure free operation for a specific duration under a specific environment [15]. Studies are conducted on various aspects of software engineering to analyze and infer results regarding the quality of software and estimation of costs and benefits involved.

A study was conducted on reliability measurement and modeling for large commercial software by observing and analyzing the trends in test execution data that was collected during system testing [15]. It is a comparative study on various reliability measurements and modeling tools available for use in large-scale commercial software environment. The work concluded that run based models are robust and provide good measures of reliability if runs are conducted homogeneously across the test cases and over a certain period of time. The data used was coarse-grained and thus the testing environment was not very conducive for robustness of Execution-time-based models.

A major part of software engineering research deals with the cost effectiveness of various methodologies and procedures. For any software, the cost of development and the cost incurred in its testing and maintenance is an important concern to be considered. Walter J Gutjhar proposed a generalization of the input-domain based software reliability measures by including the expected failure cost under the operational distribution as a measure of unreliability [16]. The principle is in accordance with the recent attempt in

software engineering to use risk as a structuring element in software development. The model used completely random selection strategy for risk estimation. This work is of significance for software metrics study too.

These studies and others works undertaken in similar areas emphasize the complexity of a software development project. The tasks involve deep knowledge of modern computing methodologies and a good understanding of advanced techniques to deal with design and implementation issues.

There have not been many serious studies specifically focused on programming contests and about the research that can be initiated from such events. This is mainly due to the lack of involvement from the industrial consortium. The programming contests are usually events conducted in limited domain for no scientific reason of doing software studies. Though not viewed as a very important issue currently, programming contests have their own patrons and backing from the development world.

An application called Smart Security Manager has been designed for executing “unsafe” code especially in the context of programming contests. The study of need for such a system, the approaches considered and the implementation details of the final package have been recently published [10].

The JC software attempts to efficiently use the current technology for the purpose of conducting programming contests in a secure and safe environment. The study of various contests currently being conducted provided the necessary knowledge for deciding the design considerations and system issues involved. The software engineering studies have brought into focus the scope of using programming contests as a forum for analysis of software, study of failure rates and the study and analysis of coding practices.

CHAPTER 4

DESIGN CONSIDERATIONS

4.1 Software Evolution

The construction of software is an engineering task that involves much expertise, planning and effort as in any other physical construction project. A software life cycle typically has a number of phases that are followed for successful completion of the entire project. The process model in Figure 4.1 shows the main classification of tasks on a software project cycle. Though the phases are depicted in sequential order, this is not always the case. This figure is a simplified view of a software life cycle. There is no strict linear progression from the first to the final phase.

4.1.1 Phases in Software Lifecycle

According to Scacchi, A software life cycle model is a descriptive or prescriptive characterization of software evolution [2]. This view suggests that software process models can be intuitive or very system specific. The Requirements Analysis phase involves defining and getting a complete description of the problem. The feasibility study, as a part of requirement analysis, involves the assessment of the problem to see if an economical and technically feasible solution is possible.

The Specification is the act of identifying and formalizing the scope of solution, the attributes and relationships of the objects along with the system constraints. This leads to the division of the problem solution domain into manageable pieces called modules to denote logical sub units.

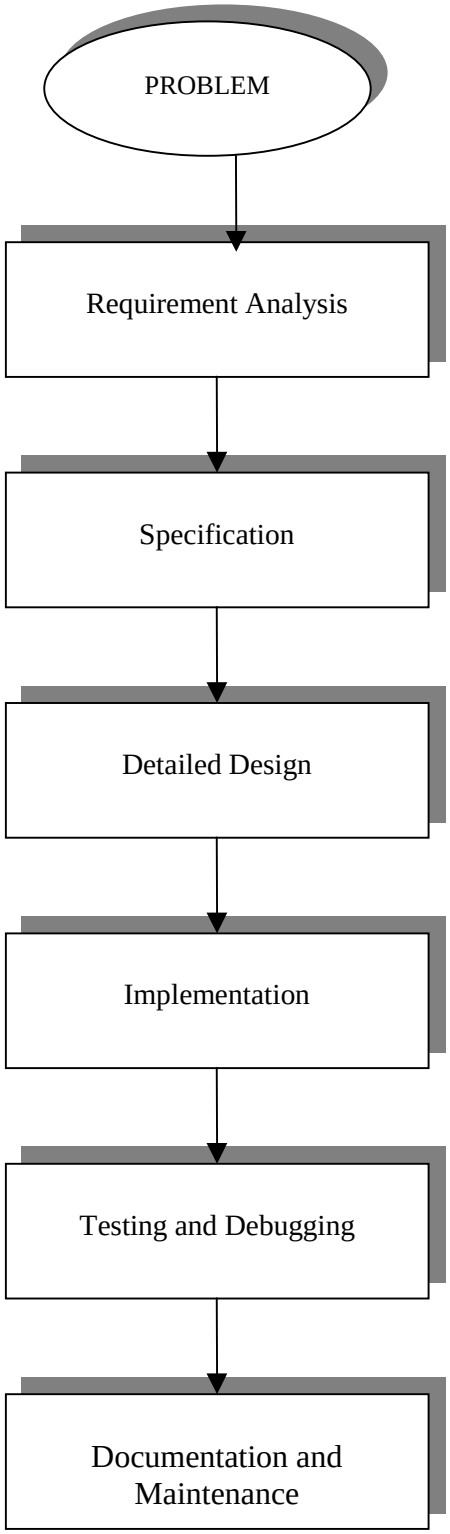

Figure 4.1: Simplified View of Software Development

The Design phase is an important step as it is the prerequisite to the implementation of the proposed solution to the given problem. It includes the crucial task of separating the *what* from the *how*. A precise definition is given to the interfaces among various modules designed.

This phase is followed by the actual Implementation. Instead of directly starting to translate the design specifications into working programs, an intermediate step of developing pseudocode is introduced for better clarity and understanding of the solution technique. This phase presents a broad scope for improvement and optimization at various levels.

After the implementation, the application developed is tested for erroneous or anomalous conditions in the Test and Debug phase. This phase is practiced in two flavors, Verification, to test if transition from one phase to next is correct, and Validation to check if the work is on the right track as regards to fulfilling the requirements.

Finally, the phases of Documentation and Maintenance handle the documentation and maintenance, which includes enhancements, sustenance of usefulness, performance improvements and conversions for portability or upgrade.

4.1.2 Efforts Involved in Each Phase

As shown in figure 4.2, the relative efforts that are put in the testing phase are more than the other phases in the life cycle [1]. Table 4.1 quantifies the relative efforts put in the different phases of the software life cycle. The various phases of the life cycle and all the sub classes of the system design considerations will now be dealt with in great detail in the context of the JC software project.

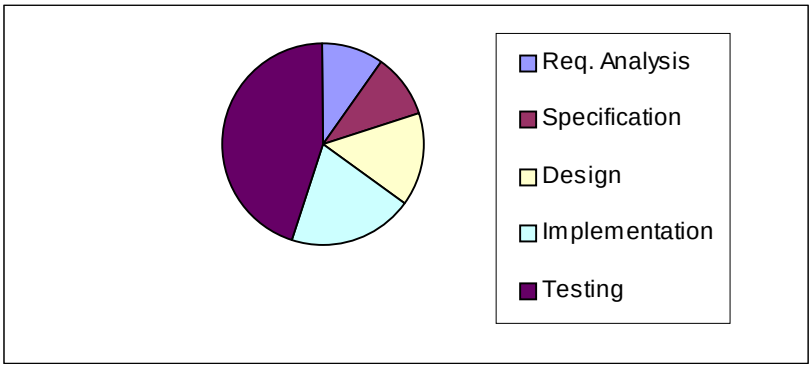

Figure 4.2: Relative effort for various activities in a software life cycle

Activity	Relative Effort
Requirement Analysis	10%
Specification	10%
Design	15%
Coding / Implementation	20%
Testing	45%

Table 4.1: Relative Effort Percentages

4.2 Java Challenge Software

The Java Challenge Software Project was a task on the same lines as any other software development project. This chapter describes all activities in the initial phases in of requirements, specifications and design details. The implementation details are explained in Chapter 5. The testing strategy and detailed test plans are described in Chapter 6.

4.2.1 Problem Description and Requirements

The main objective of the JC software project was to fulfill the need of a complete self sustained package that is an implementation of the mechanism of conducting a programming contest. The main requirements of the project were:

- The processing should be automated
- The package shall process the entries correctly as per the contest rules
- It shall not compromise the security of the platform
- Evaluation shall be done in an isolated set up with no outside interference or access
- The application performance shall be reliable and consistent
- There will be enough scope and proper provision for administrative monitoring
- It shall be suitable for World Wide Web or Electronic Mail to be the mode of entry
- Suitable data shall be collected for analysis of coding practices

These requirements are a general idea of what the JC software should attempt to fulfill. The issues mentioned here have a very broad meaning and need to be narrowed down to the true attributes before the project progresses to next stage. Some system restrictions need to be applied for a specific design.

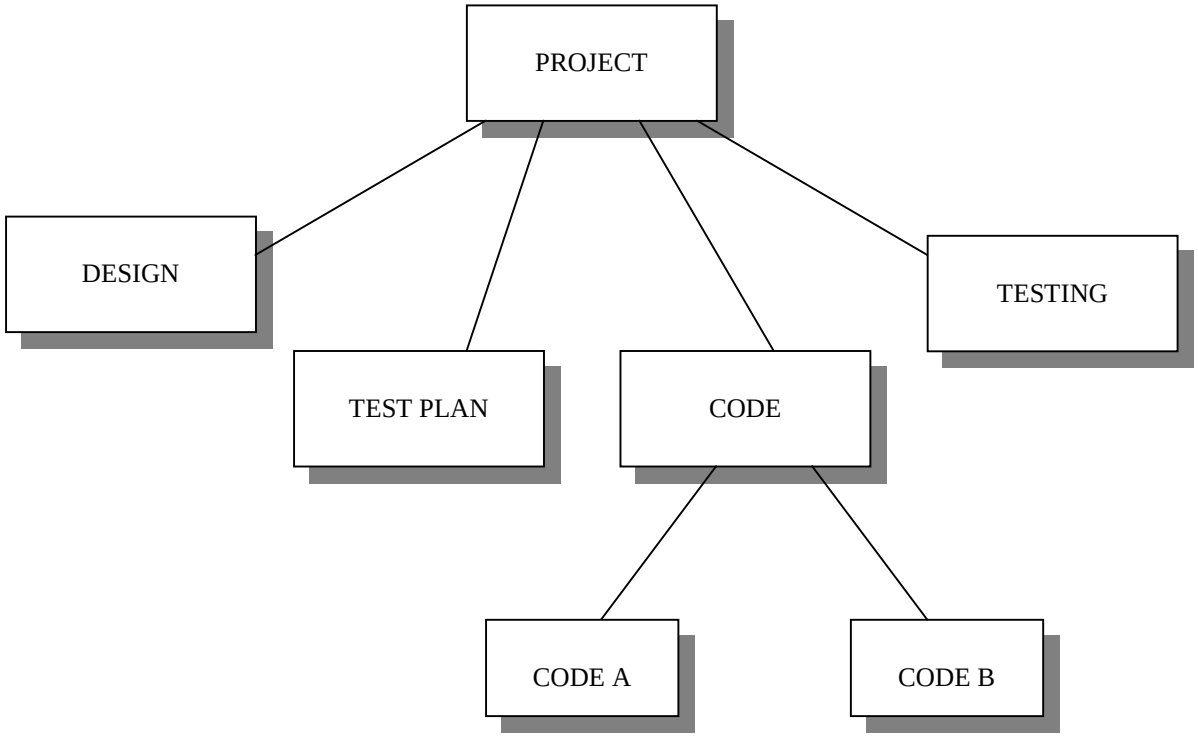

Figure 4.3: Work Breakdown Structure of a Software Project [1]

As shown in figure 4.3, the tasks at hand can be broken down into distinct modules that can be then dealt with separately. This helps in better planning and control of the entire situation. The main characteristic of the project under consideration is that it has quality constraints fixed in advance and the goal is to produce effectively a system that satisfies those quality constraints. Proper understanding of the entire project helps in separating the sequential tasks and the tasks that can be performed concurrently. This will ensure that the project progresses in a disciplined manner and there is no wastage of resources at any stage.

4.2.2 Project Specifications

Keeping all the requirements in view, the JC software project specifications were drawn to formalize what the system shall be capable of performing.

- The system shall run on a POSIX compliant platform
- The tasks shall be automated with appropriate means
- There shall be a means of authenticating the contestants' identity for security reasons
- All contest entry codes shall have restricted access to the system
- Result listings shall be available on an official JC web site
- Data collection shall be done through an Entry Form
- Data analysis shall include time performance analysis, frequency of errors, failure rate as related to contestant characteristics and related issues

These specifications narrowed the options to be considered and also informally decided the course of action that the project shall follow to achieve the proposed objectives. There still exists some scope for variation and ambiguity even after the specifications have been defined due to constraints in the availability of tools and other issues that shall emerge when these specifications are put into detailed design.

4.3 Design Options

The following sections describe the characteristics of various design options that were considered for achieving the objectives that are laid out in the specification list. The relative merits and demerits of each approach will be discussed to throw light on the decision making process in which the designs are rejected, accepted or modified for suitability and better performance in the given context.

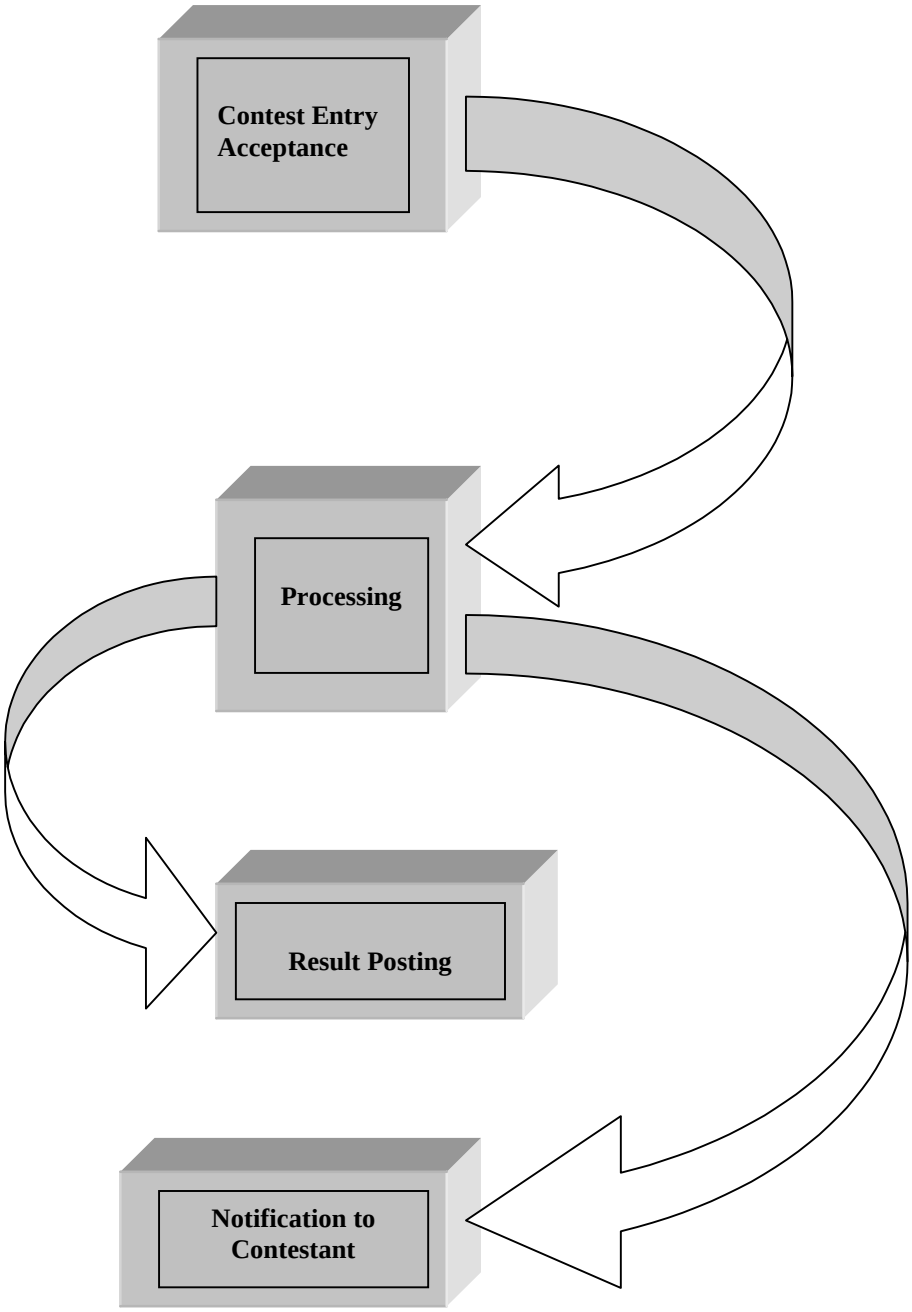

Figure 4.4: General Layout of the Java Challenge Software Project Modules

4.3.1 Contest Entry Acceptance

The first step in software that conducts a programming contest is to provide means of accepting the contestants entry. There are certain specifications that need to be prescribed for uniformity of acceptance mode and criteria. The candidate interested in contesting the challenge needs to have a means of submitting his entry for evaluation. Keeping in view the fact that the Java Challenge is an online programming contest, this task of accepting entries can be achieved in two ways:

4.3.1.1 Via World Wide Web

One of the options considered to facilitate submission of entry code was to provide an interface on the Internet. This means that the information would have to be exchanged through a special interface. As shown in figure 4.5, the contestant would develop the entry code in response to the challenge posted. This code would be transmitted through a form available on the Web. The contents of the form would be received and processed by the processing unit.

The standard mechanism that enables transfer and processing of data through the Internet is called the Common Gateway Interface (CGI). It is the standard for interfacing the external application with the information servers like HTTP or Web servers. Using this method of entry submission, the code would be sent as content of a form. The CGI script associated with the form would then process this content. This processing involves the evaluation and grading of the entry code and logging of the resulting data. The CGI script provides an efficient means of accepting data but it does not have a provision of returning static data entirely by itself.

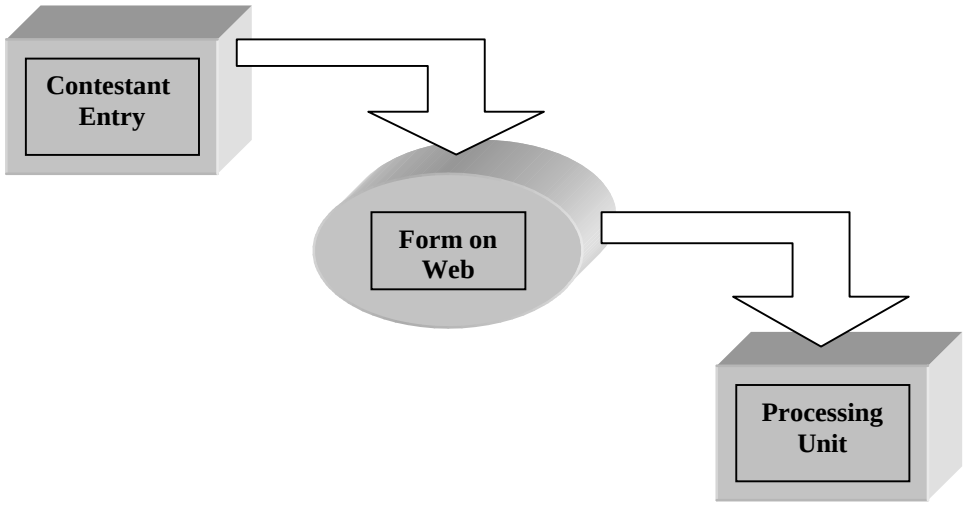

Figure 4.5: Entry Acceptance from the Web

A CGI program is executed in real time and is capable of outputting dynamic information [17]. Since a CGI program is executable, it leaves the host system open to the outside world. Thus, CGI is a security threat to a system and appropriate security precautions need to be taken while using this mechanism. The accessibility to the CGI library is restricted to control its usage on a web server. With proper permissions and access paths, the average user can execute the CGI script. Such scripts can be written in a scripting language like PERL or TCL or in conventional programming languages like C or FORTRAN.

Figure 4.6 is the schematic representation of the process of executing a CGI Script via an HTML Form.

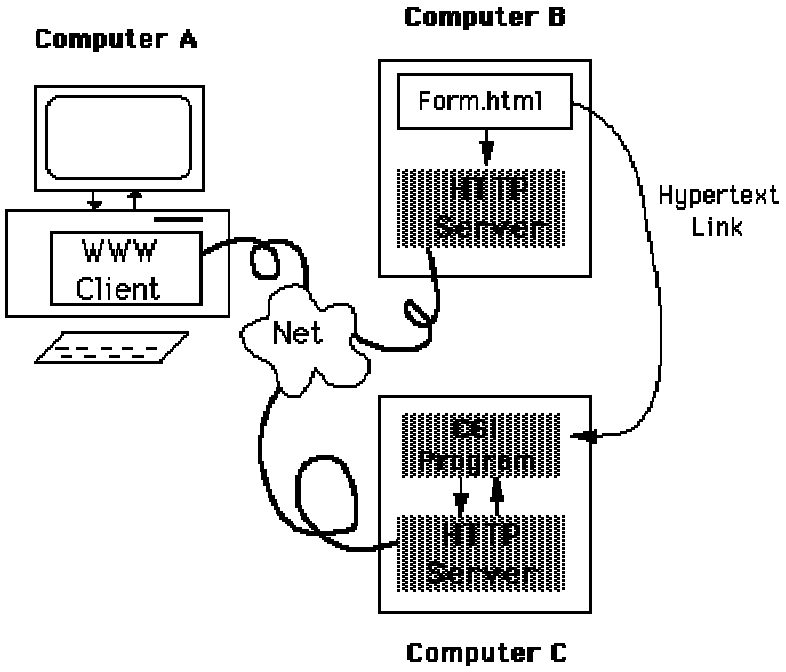

Figure 4.6: Mechanism of CGI Script Execution from HTML Form [18]

With the addition of a set of special tags in HTML code, a form can be created on the document to enable the user to input data. This form is associated with a script to be executed. This execution is triggered by a specific action on the form document. In figure 4.6, the Web client running on Computer A acquires a form from some web server running on Computer B. The form is displayed, the user enters data and the client sends the information to the HTTP server running on Computer C. There, the data is handed over to a CGI program or script which prepares a document and sends it to the client on Computer A. The client then displays the document. Instead of displaying a document in return, there can be other processes or executions triggered.

Though this method of submission seems to serve the purpose of the programming contest entry acceptance, there are other issues involved in the implementation of this design. Being a matter of system security, the CGI will have to incorporate elaborate security measures to ensure that it does not provide a security hole. The requirement of the scripts to be in a specific directory so that the Web server knows to execute them instead of displaying it needs special permission to be given by the web master or the Web server administrator.

The submission through a form on the Web leaves the system open to unauthorized and malicious actions. There is no means of authenticating the sender of the data and the identification of participants cannot be properly monitored. It would be difficult to resolve controversies regarding results as the contestant identity can be easily faked. Moreover, a CGI script must execute fast and display information soon enough because the user will click the SUBMIT button and expect to see some information immediately.

For all the reasons mentioned above and many more, this option of entry submission was not chosen for implementation.

4.3.1.2 Via Electronic Mail

Another means of transmitting data over a network is by use of the Electronic Mail facility. Figure 4.7 shows the basic transfer of content according to this method. This option requires that the sender and receiver have the facility of using e-mail as the mode of communication.

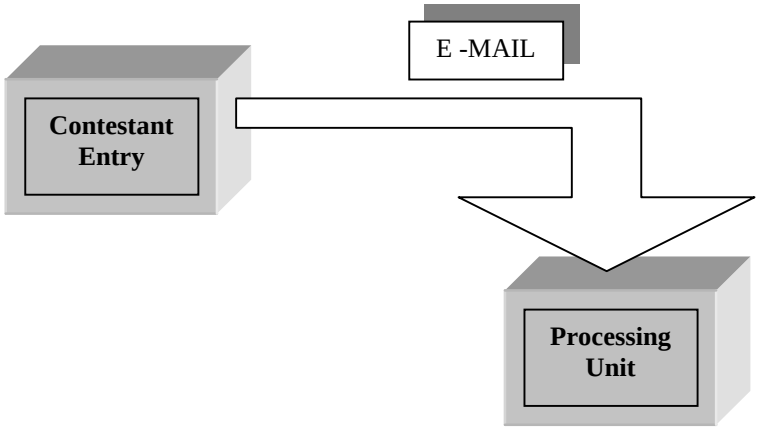

Figure 4.7: Entry Acceptance via E-Mail

E-mail is the transfer of content across network with the help of Mail Transfer Agents. There are different kinds of MTAs available that perform well in specific conditions. Every system that is networked and facilitates transactions usually has some kind of MTA for e-mail message exchanges.

On most Unix platforms, sendmail is available as the MTA. This is a package that transfers mail messages across by the Simple Mail Transfer Protocol. This is an efficient system that works well and performs fast but poses security problems, as it is not designed for high security. Sendmail on unix can be used with the available mail managers as Pine or Elm.

An alternative to sendmail is qmail. Qmail is an Internet Mail Transfer Agent (MTA) for UNIX-like operating systems. It's a drop-in replacement for the *Sendmail* system provided with UNIX operating systems [19]. *qmail* uses the Simple Mail Transfer Protocol (SMTP) to exchange messages with MTA's on other systems. Qmail has a few

advantages over sendmail as it is designed for higher security. It provides flexible interfaces for the mailboxes and is also simple to configure and use. *qmail* parallelizes mail delivery for faster processing. It also supports a new mailbox format that works reliably even over NFS without locking.

After an e-mail message is received, it needs to be processed and filtered before the contents can be used for other phases of the software project. The option that was considered for e-mail filtering was to use Procmail. Procmail has a mail filter that can be customized for any desired action to be performed on the arrival of a new message. It provides a means of filtering incoming mail and processing the contents as required. With proper invocation, this filter tests the message content and does the prescribed operation. Procmail is a powerful filtering mechanism that can be used effectively not only to filter e-mails but also to perform complex processes and executions while it is activated.

The availability of sendmail on Unix platform and the feasibility of determining the source of every message were desirable qualities in the system. E-mail transactions can be easily traced to their source and thus the contestant identity can be monitored and authenticated efficiently.

These special features available in Procmail that are suitable for the task at hand prompted the selection of this e-mail and filter combination to be used as the means of accepting contest entries and communicating with the contestant in an efficient manner. The usage of procmail and the functions of the filter shall be discussed in later sections of the thesis.

4.3.2 Entry Code Processing

After an entry has been received in response to the challenge, the next step is to evaluate the code and determine if it qualifies to be among the winning entries. The evaluation of code mainly consists of compiling the code, executing it and testing if the output is as expected. This processing of the entry code can be done in two different kinds of set ups that both satisfy the requirement of being independent of the rest of the system.

4.3.2.1 Evaluation in Physical Isolation:

To ensure that the evaluation of all entries is fair and done in a consistent manner, the simplest setting would be that transfers the entry code to an independent isolated system and conducts the processing there. For this setting, a new system would be required to serve as the processing platform for the challenge.

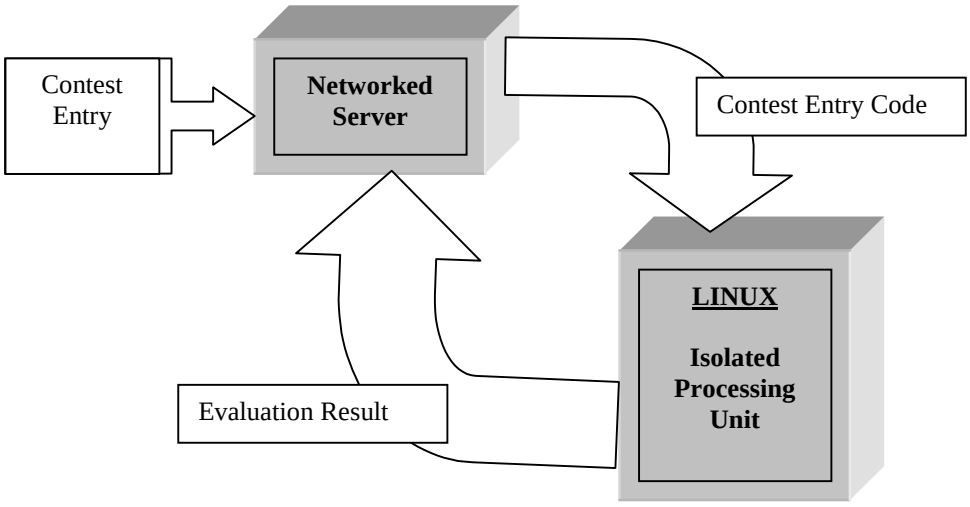

Figure 4.8: Evaluation in Separate Setup

A networked server receives the entry code via e-mail. This code is then transferred to a different system (probably Linux) where it is processed and the results are then sent back to the parent system. This set up requires an additional server that needs to be configured properly to match the parent system settings. This option was considerably well formed because of all the flexibility it provided. The processing could be done independent of the rest of the package. The system would automatically get rid of any illegal code by trapping the errors. The presence of an isolated environment for processing of entries would ensure that there is no discrepancy in the procedures and eliminate the possibility of usage of unfair means.

Though so advantageous, this setting is not a very practically feasible and economical option. It involves the added overhead of configuring and maintaining the additional system for the processing work. This setup would have to be somehow connected to the parent server for transfer of code files and result data. If not networked, these tasks would have to be done manually which would fail the objective having a truly automated system for running the programming contest.

The data would be distributed all over and there would again have to be duplication of work for transfer and archiving of all data. This option was partially implemented in view of the secure environment it provided but the attempt was not successfully concluded due to administrative obstacles. Moreover, current version of Linux (REDHAT 5.2) does not have a compatible version of jdk1.2, which made the whole effort futile.

For these reasons, this option was abandoned after attempting to set up the Linux server on an old i486 machine (contest.csee.wvu.edu).

4.3.2.2 Evaluation in Isolation

This design option was a modified version of the one mentioned above. It eliminated the need of creating and maintaining a separate system for the processing tasks.

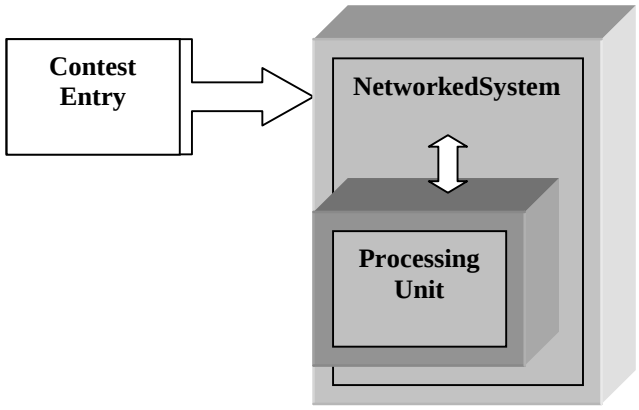

Figure 4.9: Evaluation in Isolation

As shown in figure 4.9, this method of evaluation does not need an additional system. The processing is done on the same system that receives the contest entry. This implies that the processing unit is isolated from the rest of the system and packages by means of appropriate partition walls in the form of a closed shell that does not interact with the outside world.

This system setting is possible only in cases where the entire process is running under the root or administrator. The ability to isolate a portion of the system setting can be achieved only by commands that are accessible at the system administration level. To be able to run at user level, this mode of processing has been modified. Further description is available in later chapters.

4.3.3 Authentication of Contestant Identity

While the entries are being processed, there is a need for determining the identity of the contestant and to ensure that there is no cheating or forgery involved in sending the entries in response to the challenge. There are various ways of determining and stabilizing the entry options in a system of programming contests. The following were considered to establish identity of the contestant and provide authenticated participation.

4.3.3.1 Password Protection

One of the oldest and most common means of controlling access and monitoring usage of a facility is to make it password protected. This means that the system or facility would be available for usage only when a login name and password successfully authenticate the identity of the requesting party.

In the context of conducting a programming contest as the Java Challenge, this strategy requires issue of login and passwords to the parties interested in participating. The contestant who desires to send in an entry applies and requests a login and password. These are stored in the system database and are referred to every time the contestant tries to access information or send a new entry to the challenge.

Though feasible and fulfilling, this option is not a very appropriate means simply because a programming contest is not of any special strategic importance to be password protected. Creating and enabling login-password information usage would incur tremendous overhead on the system maintenance. It would additionally give rise to the need of giving the users the flexibility of configuring their details and modifying the password at their will. This was thus not chosen for implementation.

4.3.3.2 Encryption Keys

Another sophisticated means of ensuring secrecy and privacy of the user on a system is to use cryptography. A public key cryptosystem can be used to successfully encrypt messages sent between two parties such that no third party will be able to decode and understand them [20]. The encryption mechanism can also be used to create a digital signature that is unique to the owner and thus authenticates his identity.

When a contestant sends an entry to the Java Challenge, the message can be encrypted to ensure that it cannot be seen and used by any other party midway. The contestant can be provided with a key pair that shall be used to create a signature that will always accompany the entry he sends. This is an effective means of preventing fake identity and ensuring that the information exchange is completely secure and private.

In a public key cryptosystem, each participant has a pair of keys, a public key and a private key. The public key is available for anyone to look at but the corresponding private key is a secret kept only by the owner. Both the keys specify functions to be applied to the message. These functions for any participant are a "matched pair" in that they are inverse of each other. This means that any message if transformed by both the keys of the pair consecutively, in any order, would yield the same message back.

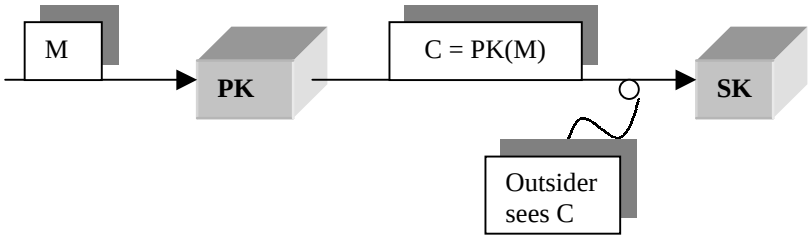

Figure 4.10: Message Encryption Mechanism

When applied to digital signatures, the encryption provides both authentication of the signer's identity and the authenticity of the contents of the signed message. The entire system of cryptography works on the assumption that the secret key cannot be determined or computed by anyone except the owner in practical time duration. The security of communications relies on two factors: the secrecy of the decryption key and the difficulty of computing or finding it without knowing it.

This is a perfect tool for electronic communication that needs to be authenticated. From figure 4.10, the major difficulty of designing a workable public-key cryptosystem is in figuring out how to create a system in which the transformation PK (public key) for a messages (M) can be revealed without thereby revealing the means of computing the corresponding inverse transformation of the secret/private key (SK).

The complexity of the whole procedure and all the political controversy regarding how difficult can an encryption mechanism be were enough reasons for leaving this option as a research topic currently not suitable for implementation in the Java Challenge software project.

4.3.4 System Security

A major issue to be considered when external code is run on a system is security. A strong solution is sought to ensure that the applications from the contestants are run on the system without putting the system at risk and prohibiting all illegal pathways to system resource access. There are many ways of restricting access to the system while a code is executing. The choice depends mostly on the extent of security desired and the level of usage of the available tools.

4.3.4.1 Chrooted Environment

The most used mechanism of high level security on Unix environments is the Chroot facility. Chroot is the command that allows the system administrator to force a program to run under a sub set of the file system, without allowing access from it to any other part of the file system.

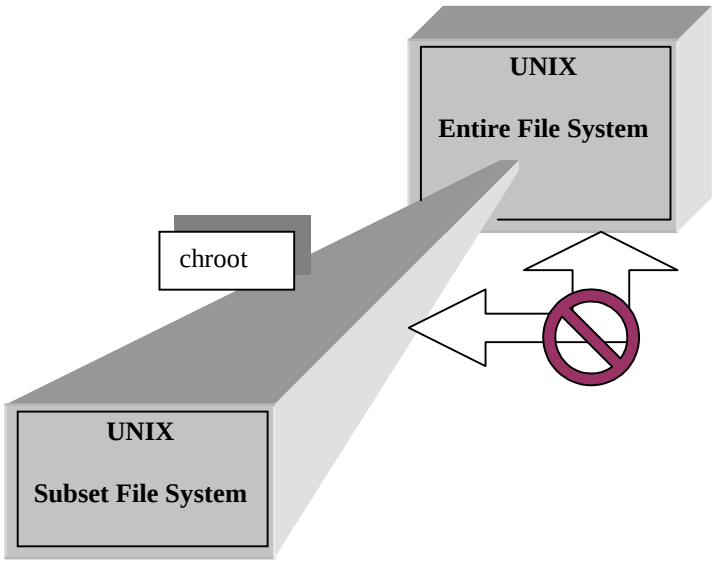

Figure 4.11: Chroot Environment Setup

As shown in the figure above, the chrooted environment is a part of the entire file system but does not have any access to the rest of the system. This system is most often used for anonymous ftp from a server and for the server web tree.

This command completely fulfills the security requirement of the JC software but cannot be implemented at any other level except the root on the system. The implementation problems of this design include the complexity of deciding if it is a doable task. If the subset file system is to be complete in itself with the top level as root, extensive efforts are needed to duplicate the real tree structure without the real contents.

If the tree is very wide spread with links and scripts reaching out all across, chrooting is not a very efficient solution.

Chroot provides an additional security layer but is not so completely impenetrable. Any program, if running with the root privileges, can break out of the chroot trap. Any user who attempts to do this simply requires access to a C compiler or Perl interpreter and security holes to gain root access. Also, due to the limitations of Unix security architecture, the server cannot run as more than one user with the current server model while maintaining sufficient security on other fronts.

Due to the above mentioned limitations and other system administrative difficulties, the system security problem was not solved by using the chroot method.

4.3.4.2 Sandbox Technique

The problem of system security is solvable not only by redefining the run time environment of the system but also by specifically restricting access to resources by creation of a new control class. As Java Challenge is based on Java programming languages, one choice was to define a separate accessibility class that allowed certain resource access and prevented the rest by throwing appropriate exceptions.

This design choice is aptly suitable for the purpose as it clearly controls the system access in a defined manner. Such a class and its associated methods would eliminate any security hole issues and remove all ambiguity regarding the environment in which the code would run. This setup would work by forcing all calls from the application code to pass through this accessibility class. The request is fulfilled if the method is permitted, otherwise it would be rejected as a security exception.

This method of controlled access is a valid solution and is also implemented in the jdk1.1 security model with some more complexity. For the task at hand, it is not very practical. In view of the vast number of classes and methods available in Java, it is impossible to correctly bring together the necessary subset. The class hierarchy that governs the entire class library dictates the need of the super class for any functionality to be enabled by the sub class. This would blow the accessibility class out of proportion and thus render it ineffective in solving the security problem.

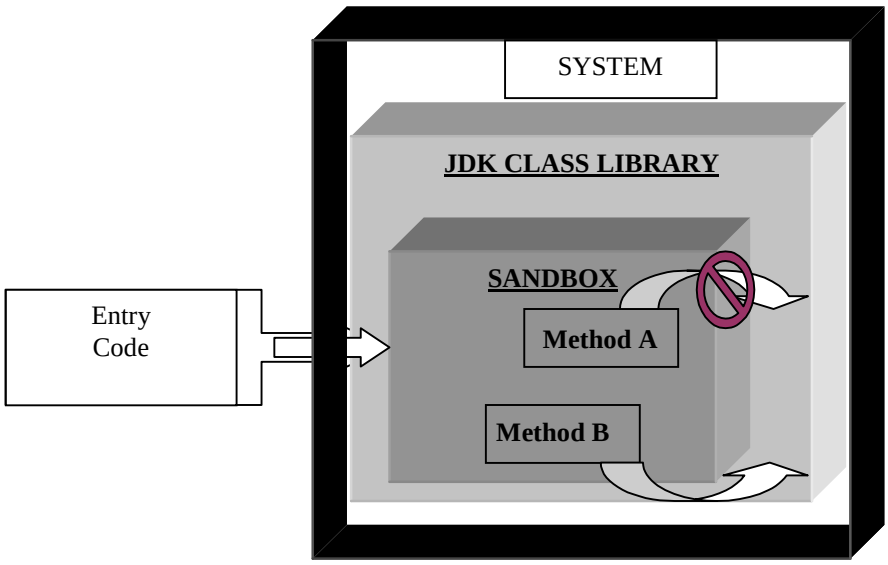

Figure 4.12: Mechanism of Sandbox Technique

Figure 4.12 shows the schematic view of the functioning of the sandbox method. If the entry code arrives, it is sent directly to the sandbox, which has a number of methods and sub classes too. If the program calls method A, which is prohibited or

restricted, the program shall throw an exception but if a call is made to method B, the execution will proceed smoothly as method B is allowed access and execution.

4.3.4.3 Java Security Manager

In the context of conducting the Java Challenge programming contest, Java 1.2 has all the features required to enforce and control limited access to the system on which the application runs. The security manager in jdk enables the configuration of system permissions to be applied when an application or an applet runs on the system.

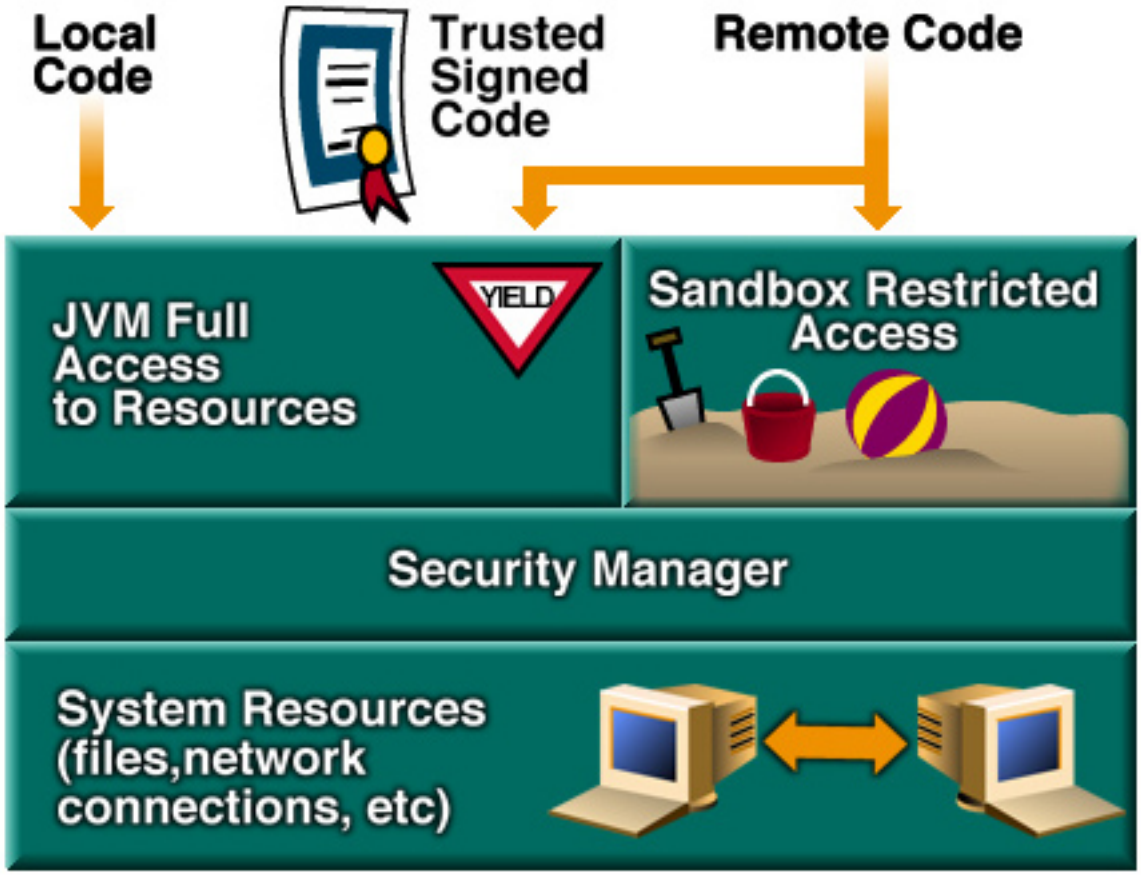

Figure 4.13: Java Security Model for jdk1.1

The java security architecture has undergone tremendous improvement and enhancement [21]. The original security model also called the “sandbox” model provided a very restricted environment for running untrusted code obtained from the open network. According to this model, any code that was local had complete access to all vital system resources where as remote code (as an applet from the web) could access only limited resources present inside the sandbox.

The latest release of Java, the Java 2 SDK, introduces new concepts of permission and policy that enable fine grained, highly configurable, flexible and extensible access control. It also provides tools for cryptographic services and certificate and key management classes and interfaces. Figure 4.13 shows the working of the security model.

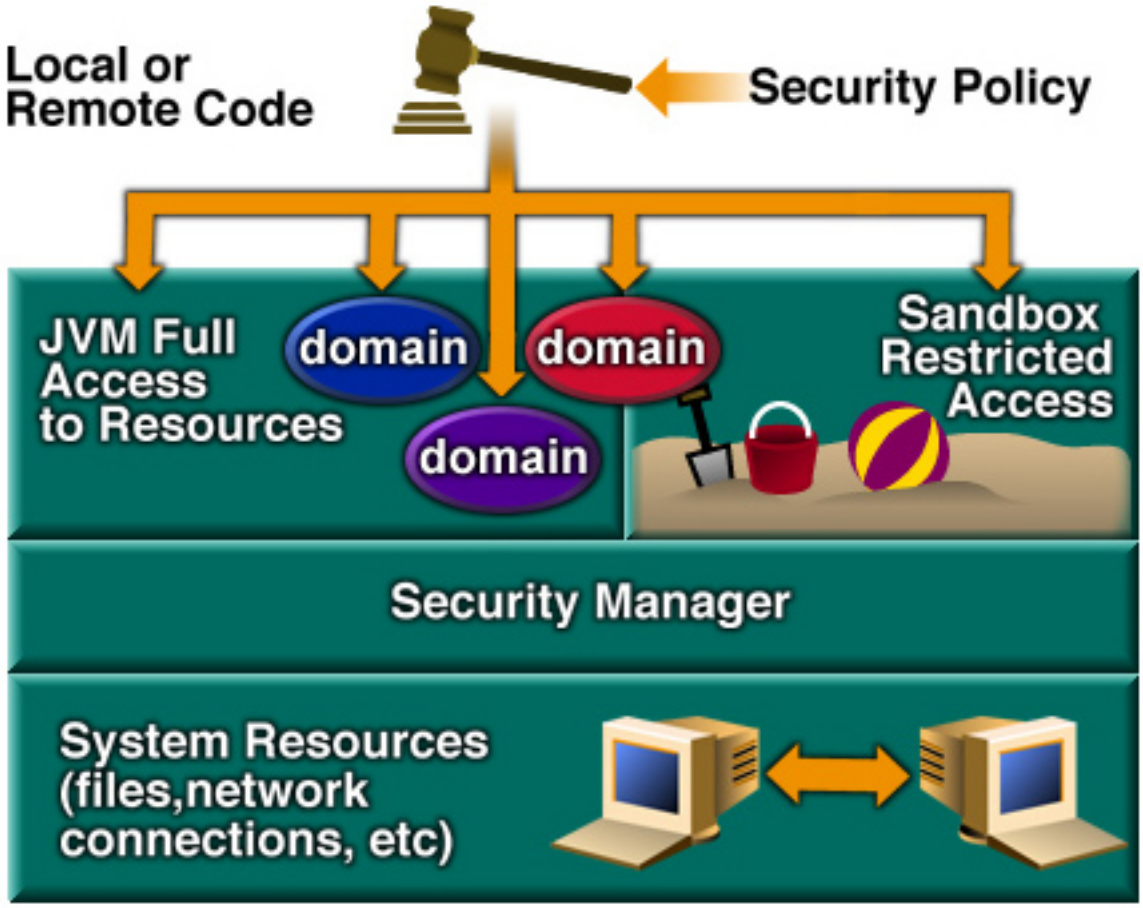

Figure 4.14: Java Security Model for jdk 1.2

With the concept of “signed applet” already in implementation, the later versions of Java attempted to enhance the security manager by enabling all code, whether local or remote, to be subject to the security policy.

The security policy defines a set of permissions available to code that can be customized and configured by the user. With the current architecture of the security model, applets can still run in the restricted sandbox environment and the applications can be optionally subject to security policy or given complete access as in earlier versions. The domain grouping has now changed from full access and sandbox access to a broader range to include intermediate domains that have less than full access but more than the sandbox.

Along with the file access permission enhancements, extensions have been implemented in the cryptography architecture too. The Java Cryptography Architecture is a framework for accessing and developing cryptographic functionality on the Java platform. Certificate interfaces and classes for parsing and managing certificates have been introduced in jdk 1.2 for better security features.

After thorough research of the features provided by the Java Security Manager, it was found to satisfy all the security configuration requirements of the Java Challenge Software. It was then implemented and used appropriately. The usage of the security features of Java ensured that the entry code running on the system was fully under control. Access to the file system, the network and to all other vital resources was definable and configurable as desired. So, this design option was chosen for handling the system security issue.

4.3.5 Automation Techniques

The main advantage of any processing software lies in its ability to perform functions at specified times without the user having to physically start and monitor the processes. A function can be automated by giving it a time handle or arranging for an event trigger that starts when a certain condition is fulfilled or a particular event occur.

One of the objectives of the Java Challenge software project is to create a package that performs all the tasks in an automated manner without the need for any manual intervention. The requirement is basically to accept entries and process them immediately without having to wait for the administrator to check if an entry has arrived and then start the processing scripts. Apart from constant polling for new entries which would put a lot of load on the server, There are a number of ways to achieve this objective in the context of Java Challenge software.

4.3.5.1 Cron Job

One of the features of Unix is the provision of timer and time dependent functions. Crontab is a program that runs in conjunction with the Unix shell that functions as an event timer. Cron command used to start a process that executes commands at specified dates and times. A cron job can be submitted at any level on the file system and it shall execute given commands at specified times. These jobs are used commonly for regular tasks in system administration. It can be configured for emptying temporary directory contents, loading processes automatically and switching log files at certain time and dates.

This mechanism was considered appropriate and also implemented to some extent but was later abandoned. The cron mechanism functioning is not very predictable and consistent when the tasks make assumptions and depend on shell and environment properties. As the cron job runs when the user is not logged on to the system, it is difficult to monitor and control. The job runs at root level and there are many ambiguities in the way it handles temporary files. A job given to cron gets triggered at specified time irrespective of the status of the previous run. This can give rise to system overload and slowdown if one of the runs is not successfully completed. For these reasons, an alternate means of automation was researched.

4.3.5.2 Mail Filter Trigger

To avoid the inconsistencies of processing and also eliminate the need for constant polling, the mail filter features were more deeply researched. As evident from experience of other developers and documentation details, a mail filter is more than a mere message-redirecting application. A mail filter has extensive features for customized configuration, redirecting of mails, automatic deletion of messages, auto-reply mechanism and also support for executing commands in the event of the incoming mail property matching the desired characteristic.

This feature is available in procmail and was a viable option but the commands that can be executed this way have some system restrictions. The command must be on the same file system. A script or command on a different server or system cannot be invoked from the e-mail filter. This was a limitation that could not be worked around in the Java Challenge software as the processing was being done on a different system that

had a mirror image of the file system on the machine that actually receives the contest entry as an e-mail message. The triggering of processing scripts by the entry of a new contest entry would be true automation. This would eliminate all unnecessary executions when there are no new entries to be processed. Due to system limitations, this option could not be pursued in real time.

4.3.5.3 Daemon Script

There are a number of mechanisms for performing certain tasks at timed intervals. The selection of the method depends on the task at hand, how accurately timed it needs to be and the priority of the job. There are ways of invoking time and event triggered processes by complicated implementation in conventional programming languages like C. These implementations can be extended to invoke jobs on different systems by calling appropriate methods with the right accessibility options.

For the Java Challenge software project, the need is to invoke an event that executes the processing scripts to evaluate the contest entries. This execution needs to be done at timed intervals with no dependence on the user's presence on the system. This process shall be responsible for evaluation of the contest entry code, logging of data, notification to contestants and updating of the result listing.

An efficient way of performing any function on a Unix platform is the use of Shell scripts. A shell script is a sequence of commands that can be executed at one point without having to issue each one of those at the command prompt separately. There are different kinds of shells in a Unix system, C shell, Bourne shell, R shell and Z shell. Each shell has its own special features and syntax for setting of global values. Though having

subtle differences, all shells basically perform the same functions in the same manner. It is a matter of considering specialized purposes and convenience to select a particular shell for use in the entire process.

A daemon script is a script, shell or otherwise, that does not terminate when the user logs off. This is a script that runs infinitely on the system as long as the server is up and alive. Daemons are usually designed for the web server, mail purposes and ftp usage. These are functions that need to be simply active and running all the time. Any regular script can be converted into a daemon by appropriate commands and run time property configurations.

This option was chosen for the Java Challenge project as it is feasible to implement and has good logging facility for monitoring and control. The implementation details of this method are discussed in the next chapter.

Now that the available options for various tasks have been considered, researched and the choices made, the next step is to implement the selected options to create a functional package that can be tested and then used.

CHAPTER 5

SYSTEM IMPLEMENTATION

In the life cycle of a software project, the design phase is followed by implementation phase. After all options have been considered, all methods analyzed and the merits and demerits of each one weighed carefully, final decisions are made regarding the choice of design methodology. Work then proceeds to bring these choices to real application platform by implementing them as part of the development code.

Figure 5.1 depicts the complete flow of control in the Java Software. The entire process starts with the entry sent by the contestant. The rules of the contest and the current challenge are posted on the web. Interested parties, who visit the site, are expected to observe all the rules for participation and qualification. The entry form is also available on the web site. The web site also holds the data regarding previous challenges held, if any. Contestants work on the problem posed and send the candidate solution as part of an e-mail message that is formatted as per the rules of the contest. Now, the Java Challenge software takes over.

This mail message passes through the mail filter and the body of the message is separated as the entry code. The processing unit for evaluation then picks up this code. The processing unit attempts to compile and execute the entry code. All processing data and results are archived for later use. After evaluating the entry, the processing unit sends out the final results for web posting and also sends e-mail notification to the sender of the entry code message. All though this process, data is archived and log files are generated for future reference and analysis.

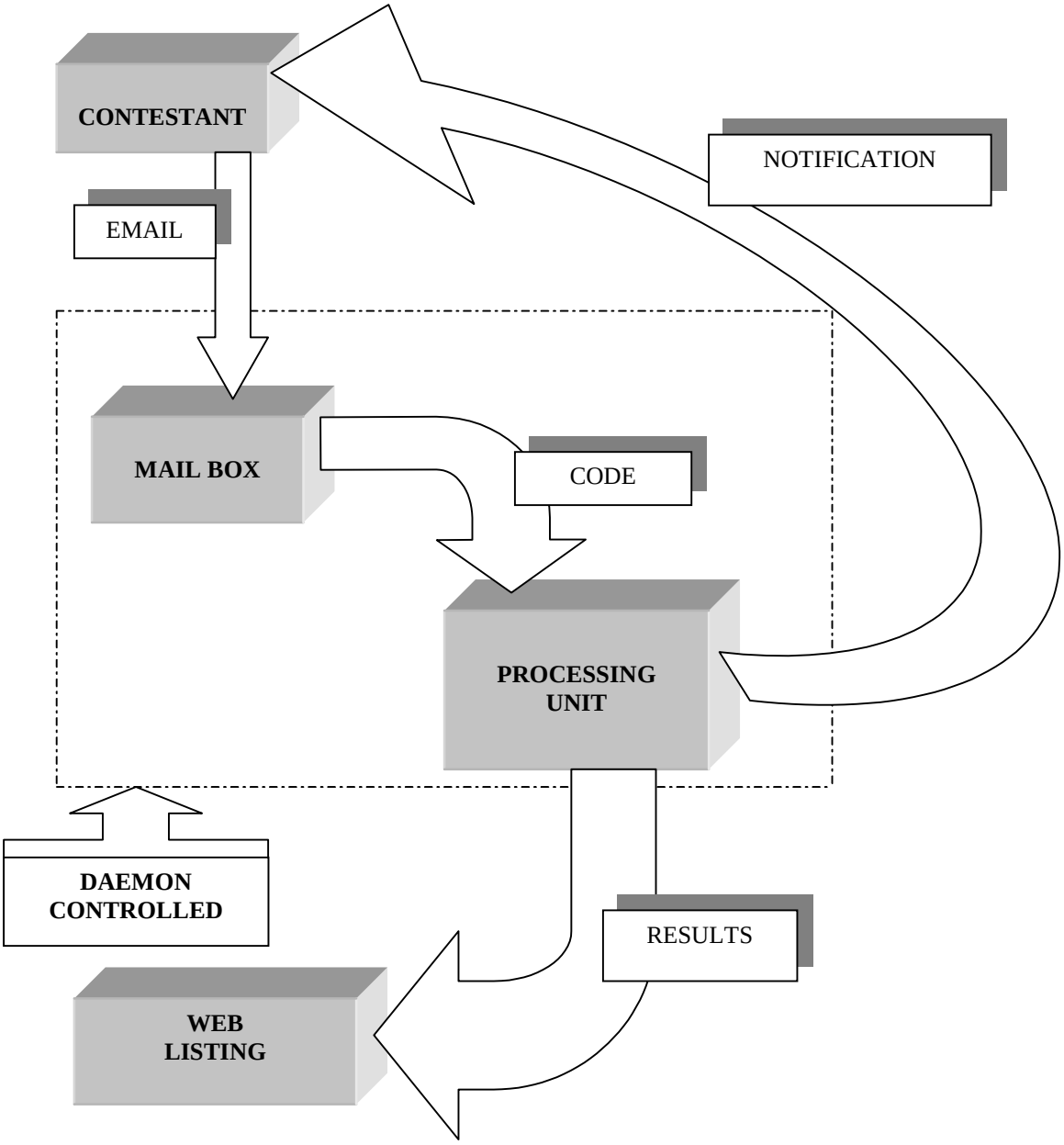

Figure 5.1: Flow of Control in the Java Software Implementation

5.1 Mail Filter

The first module that gets activated when the Java Challenge software runs is the e-mail filter. A procmail filter has been created and customized for the desired functions. The filter scans all messages that enter the system and looks for the key word match on the Subject line of the message header. On success, the filter performs the specified actions.

The procmail filter can automatically process incoming mails depending on the headers and message content. A procmail filter is usually invoked from the *.forward* file mechanism that confirms the presence and location of the filter file. The rc file used for procmail basically has a mixture of environment variable assignments and recipes. A recipe is a simple one line regular expression that is searched for in the message header. When the first match is found, the mail is processed according to the action part of the recipe.

The action lines on the recipe can be simple commands to redirect or bounce the message to specified places and can also be complex commands that execute shell scripts and manipulate the message content as desired. The action line can be a single statement or a nested block of commands to be executed in sequence. There are specific syntax formats to be followed for specifying the recipe for a procmail filter.

In the Java Challenge software, the procmail filter looks for the string *(CONTEST)* in the subject line, and if found sends an acknowledgement message to the sender. It then proceeds to truncate the mail header while retaining only the sender's address. The content of the mail message along with the sender's e-mail address is written to a file named based on the current process number and time of receipt with a .java extension.

This file is then ready for the rest of processing software. This mail filter need not be manually activated. By appropriate setting of the *.forward* file, this filter is always invoked when a new message arrives in the system. After the mail message has been processed, a copy is still retained in a specific folder in the user mailbox. Figure 5.2 is the schematic layout of the working of the procmail filter as described above.

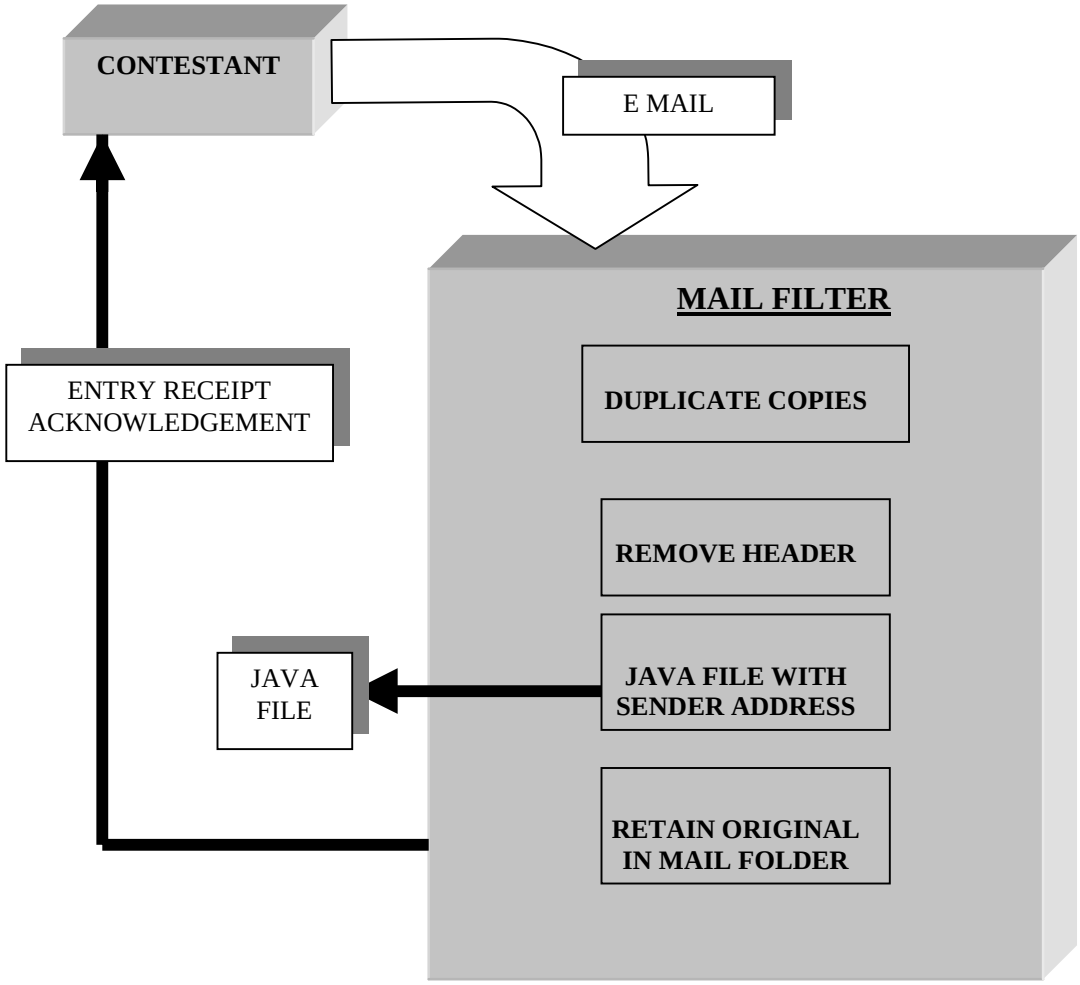

Figure 5.2: Mail Filter Mechanism

5.2 Processing of Entry Code

After the mail filter has done the work of processing the incoming message and creating the java file, the system is ready for the processing of this entry code. The processing scripts developed in Perl are invoked and they perform the functions of evaluating the code.

Perl is a scripting language for easy manipulation of text, files and processes [22]. It provides sophisticated operators and functions that have the ability to process multiple items with a single command. Every operation in a perl script is evaluated in a context and the behavior of the object depends on the requirement of the context. This language provides advanced functions for system interfacing and manipulation. Perl is efficient and faster in performance simply because it is an interpretative language.

The processing scripts for Java Challenge software are written in Perl. Appendix A contains a listing of these scripts for reference. The script polls the directory and looks for the file name of a specified format to be picked up for evaluation. The content of the file is separated into java source code and sender's e-mail address. The java code is then compiled using jdk1.2 on Solaris 2.6 platform. This java runtime environment installation is specific to the JC software access and does not interact with the rest of the system software on the Solaris.

On successful compilation, the script attempts to execute the class file generated and stores the resulting output in a file. This output is compared with the standard output or the correct output to the posed challenge to determine if the entry is correct. While, the class file is being executed, the Java security manager is invoked with appropriate arguments to enforce the permissions on the execution. This activates the security feature

of the software and checks for permissions before performing any action based on the execution of the entry code.

The processing unit performs all these functions from inside the Perl script and there are log files generated all along. Details of entry files being evaluated are stored in data files. The result of every processing action is archived for later analysis. These data files shall later be used when the coding practices are being compared with the results they yield. After the entire processing is over, the other scripts read the data files created by this script for further actions to be taken.

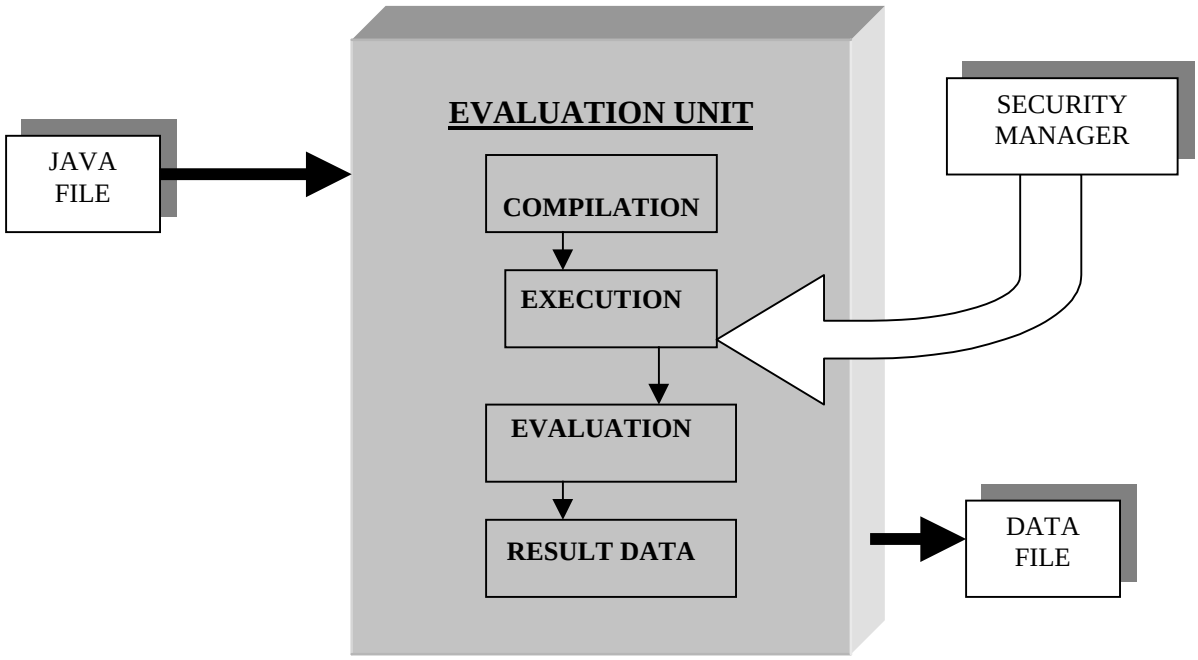

Figure 5.3: Processing Unit Operation

5.2.1 Java Security Manager

A security manager is not automatically installed when a java application runs on the system. The application has full access to all resources and cannot be restricted directly. To control and restrict the access an application ahs, the security manager needs to be specifically invoked by an extra command line parameter while using the interpreter to run a java class file. The system policy file loaded by default, grants permission to access some commonly useful properties. The default permissions can be overriden with appropriate policy file creation.

A policy configuration file contains details of what permissions are granted to code from specified source locations. It contains a list of entries that have grant entries and optional keystore entries. The Runtime environment for Java 1.2 SDK provides various graphic tools to aid in the creation of these policy files. The policytool is one such application. It can be invoked by a regular command and brings up windows with text boxes where the desired information can be entered. When finished, the tool translates all the information into a syntaxed java permission file.

The main components of specifying permissions are available with the possible values when the policy tool is used. The CodeBase is required to specify the source of the code that shall be subject to these permission policies. The optional SignedBy field is to indicate if the code has an encrypted signature for verification. The kind of permission to be granted, the target names of files or details whose access is being controlled and finally the corresponding action that is being enabled are also to be specified. When the policy file is saved and successfully loaded into the appropriate location, it can be called from anywhere for accessibility checks while running an application. Appendix C

describes the various implementation details of the java permission classes along with the syntax for the policy files. The methods available in each permission class are defined with the details of how different target names and actions need to be specified. The jar signer and keytool are additional tools available for security management. They have not been included as their functionality is beyond the scope of this project.

The policy file for the Java Challenge is configured so that any code whose source is the directory tree rooted at the level where the scripts run is subject to restrictions. The only files that have read and write permissions are the ones in /var/temp location. The socket permissions are restricted to mere listening on the 65000 port of any host. This means that the entry code does not have access to any system resources while it is being executed. It cannot connect and perform transactions over the network and cannot read or write data at any random place on the file system.

5.3 Notification Process

An application package for conducting a programming contest must not only perform evaluation and tabulation, but also incorporate means of prompt communication with the contestants regarding their entries. One way to achieve this objective is to send out e-mail message as soon as the processing and evaluation is over.

The mailing script in the Java Challenge software is designed to notify the contestants the result of their entry and update all interested parties of the current standings of the Challenge. The script looks at data files generated by the evaluation process to determine the content of the e-mail message. The logs are meant to specify if

the execution was a success and if the entry qualified among the contest winners. Sendmail on Unix is used for sending messages to the owner or sender of the entry code.

To: abc@xyz.com
Cc: aaa@csee.wvu.edu
Subject: Correct!!!
X-Status:
X-Keywords:
X-UID: 21

Dear abc@wxz.com, The following program:

```
public class contest {
        public static void main(String x[]){
        - - -
        }
}
```

is correct and has qualified! Its class size is
418 and was received on Tue Jul 6 14:06:13 US/Eastern 1999.
See http://www.csee.wvu.edu/… for current standings!

Figure 5.4: Sample E-Mail Notification

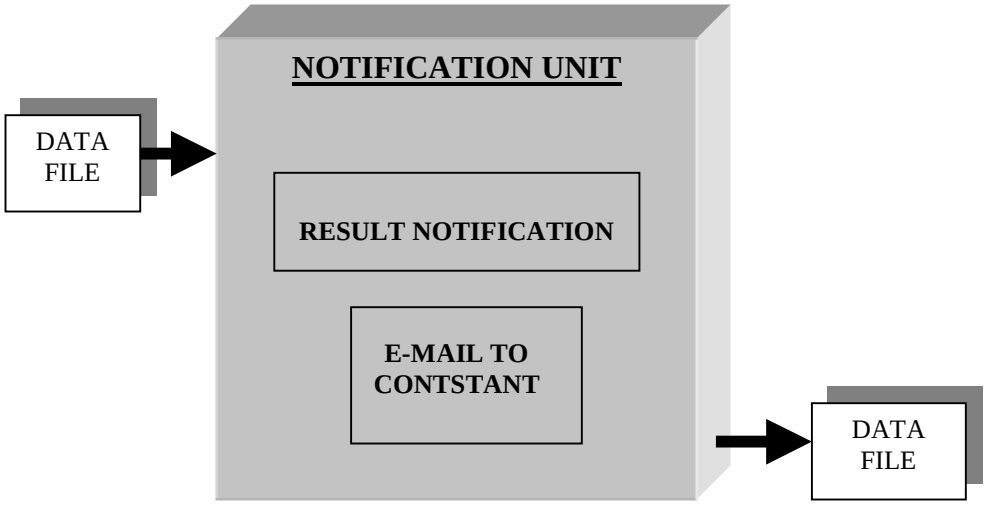

Figure 5.5: Mail Notification Mechanism

The script then sorts the data files to prepare them for the result listing. The mailing script also generates a log file that is appended every time the script gets executed. The details of this entire process can be analyzed by studying the source code listings provided in Appendix A of this thesis.

5.4 Result Listing

The final step in the entire functioning of the Java Challenge software is the maintenance of result data on the web page. The result data files that have been created by the earlier scripts need to be converted to HTML compatible table forms to be displayed on the Current Results page of JC home page.

The data files generated by the processing script and sorted by the mailing script are read through to select details of entries, which were completely successful until the end. This means the entry code that was successfully compiled, executed and produced the correct output shall qualify to be among the winning group. The eligibility of a contestant to win is dependent not only on the code evaluation but also on the requirement that the contestant fill the entry form. The status on that front is also reported on the web as a reminder. The winner is chosen as the qualified code that generated the smallest corresponding class file. In case of a tie, the time of submission becomes the deciding factor.

The web page content is automatically updated every time the script gets executed in the sequence of operations. The working scheme of web reporting task is shown in Figure 5.6. Figure 5.7 is a sample display of how the result listing is displayed on the Internet.

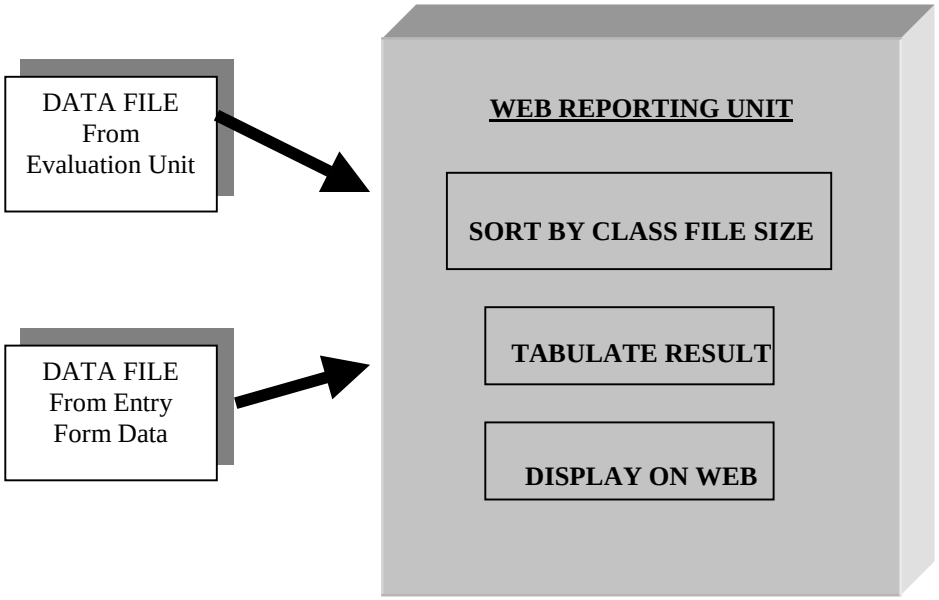

Figure 5.6: Scheme of Web Reporting

Current Standings for Java challenge

Current Leader is abc@xyz.com!!!

Rank	Size	Time Received	Person	Entry Form
1				Yes/No
2				
3				

Contest details available at http://www.csee.wvu.edu/….

Figure 5.7: Sample Result Listing on Internet

5.5 Processing of Entry Form

The Java Challenge software project is meant to provide a forum for software engineering and analysis of coding practices. To serve this purpose, the Entry Form has been introduced into the scheme. The entry form is a means of collecting real time data from contestants regarding their programming strategies and coding preferences. This data shall be analyzed later to infer correlation between various factors involved in programming tasks undertaken in a restrictive environment as a programming contest.

A CGI script has been designed for acceptance and processing of data from the entry form. The entry form is available on the JC home page and contestant is required to fill it in order to qualify in the contest and participate successfully. The form is attached to a script that executes when the form is submitted via the Internet form. The script captures the data filled in the form. The data gets written to appropriate data file, which is available to the result listing script to look at. This CGI script also creates a response page when the form is filled and submitted to acknowledge the receipt. E-mail is also sent to the address specified in the form as intimation.

It is necessary that the e-mail address specified in the form be the one used for sending the entry. This is required as the status of Entry Form Filled is determined by comparing the source of the contest entry e-mail message and the e-mail address specified in that contestant's entry form data.

The following figure 5.8 is a representation of the functioning of the CGI script as described above. A sample of the response page generated when the form is filled and submitted is shown in figure 5.9.

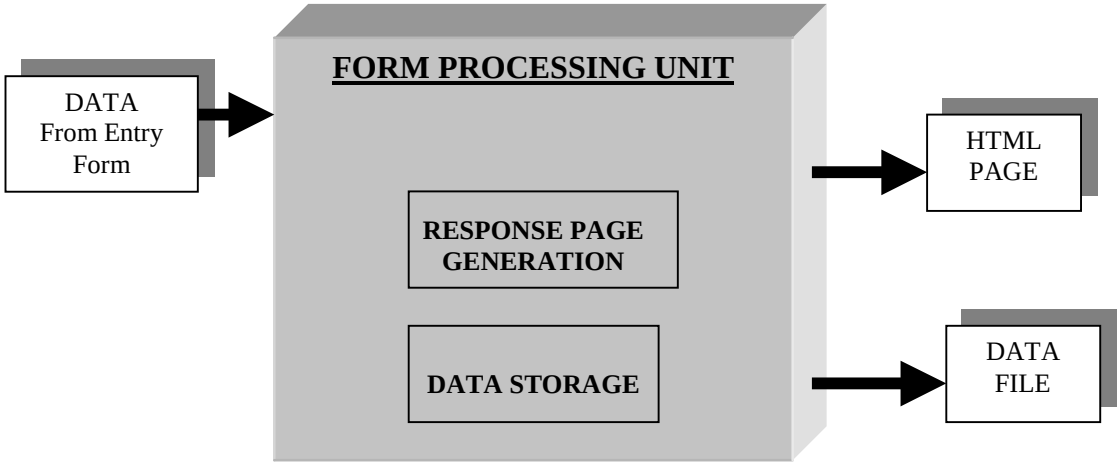

Figure 5.8: CGI Script Mechanism

Thank you for Contesting!

Thank you abc, for filling the form!
You are now eligible to contest in the Java challenge.
Remember to send the contest entry from **abc@xyz.com**
with the subject as **(CONTEST)** for successful acceptance.
Please note that entry evaluation and web posting updates happen every quarter hour.

Visit http://www.csee.wvu.edu/… for updates!!!

Figure 5.9: Sample Response Page

5.6 Daemon Script Implementation

The entire process of entry acceptance, evaluation of code and posting of results was meant to be an automated function that would not require any human intervention except for routine administration and supervision. The mechanism used for performing all functions at a periodic interval was creation of a daemon script. A daemon script is a script that goes on executing infinitely unless interrupted or killed.

SCREEN is a Unix tool for window management with VT100/ANSI terminal emulation. It is a full screen window manager that multiplexes a physical terminal between several processes or interactive shells [25]. Screen has command line options for customization and it also has a set of key bindings for various functions. Invoking screen creates a window with a shell in it and enables the creation of multiple windows with more shells, output logging, window listing, kill commands for specific windows and switching between windows. This functionality provides a means of monitoring multiple processes and running processes in a detached manner.

When a process is started on screen shell and detached, it is assigned a special identification number and shall not be terminated until reattached or killed specifically. This enables the execution of scripts or programs even when the user is logged off from the system. Unlike with the kill command in Unix, the screen processes are never left without a PID handle. This enables easier monitoring of the process, tracking its progress and proper termination when desired. Screen has been used to convert the main driver shell script of the JC software project into a daemon script so that it executes periodically without any need for user intervention.

CHAPTER 6

TESTING AND TEST CASES

The success of a software project lies in its quality. When a software project progresses beyond the implementation stage, there are a number of tasks to be performed to ensure that the application actually does what it is meant to do. Software's main objective is to perform, and perform well. There are many variations in the way quality is expressed, defined and analyzed. Different people have different perspective about the quality of software, as, a tester's view of quality is "compliance to requirements" and a user's view may be "fitness for use".

In the earlier studies of software products, a number of software system aspects were investigated which relate to software quality [24]. Quality has been defined as the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs [14]. Quality criteria can be measured at various levels subjectively and objectively. In the context of software, it is measured as the degree of correctness, reliability, usability and other similar characteristics.

There can be a decided code of quality factors and definitions that shall be used when trying to evaluate software. Though some of the factors cannot be efficiently measured quantitatively, there are ways of measuring certain characteristics or performance properties and correlating those to the subjective property under consideration. The factors being evaluated are not always independent. They overlap and might influence other properties. For instance, correctness and reliability influence each

other positively but the efficiency aspect usually has a negative impact on other factors. There are formal methodologies of managing software quality.

Quality Criteria	Definition
Product Operation	
Correctness	Does it do what I want?
Reliability	Does it do it accurately all the time?
Efficiency	Will it run on my hardware as well as it can?
Integrity	Is it secure?
Usability	Can I run it?
Product Revision	
Maintainability	Can I fix it?
Testability	Can I test it?
Flexibility	Can I change it?
Product Transition	
Portability	Will I be able to use it on another machine?
Reusability	Will I be able to reuse some of the software?
Interoperability	Will I be able to interface it with other system?

Table 6.1: Quality Factors and Definitions [23]

Before and during quality management of software, it needs to be thoroughly tested to ensure that the performance is as expected. The testing and debugging phase of a

software project begins at the test plans that are devised to test the system and bring out errors or anomalies in the software. The test plan includes various execution cases that the software shall be subjected to in order that the performance can be observed and analyzed for errors. Testing is an important phase of software development as it is here that the capabilities and limitations of the software emerge into forefront. Thorough and systematic testing is necessary to be able to rectify faults in the software and improve its performance and quality.

The Java Challenge Software has been thoroughly tested in different contexts for ensuring reliability and correctness of performance.

6.1 Mail Filter Testing

The first component in the JC software to be tested was the mail filter as the entire process starts there. Mail messages were sent in different formats to the specified address and the response was recorded. The mail filter successfully captured the right format messages but returned the ill-formed messages as invalid entries to the challenge. Here are the results of the tests conducted:

TEST	RESULT
Subject: (CONTEST)	Valid, Accepted
Subject: CONTEST	Ill-formed, Rejected
Subject: contest details	Invalid, Rejected

Table 6.2: Mail Filter Test Results

6.2 Evaluation Testing

In order to test the functioning of the evaluation process, entries were sent in at random intervals in many different formats to be worked upon by the processing unit. The correct output was fixed and the entry code was compared against it every time the execution was successful. The testing procedure also included checks for the code specifications and entry format.

While testing the processing unit, the testing of the data archiving features happened automatically as it is a part of the processing task. The data files generated were inspected for accurate recording of entry details. The log file generated by the script also aided in the testing procedure. The test cases considered and the results obtained are listed in table 6.3.

Entry Code	Compile Status	Execution Status	Output Match
Class contest, Valid code	Yes	Yes	Yes
Class Contest/Test, Valid code	No	-	-
Class contest, Exception	Yes	No	-
Class contest, Valid code, Incorrect output format	Yes	Yes	No

Table 6.3:Processing Unit Test Results

6.3 Security Testing

The JC software invokes the Java Security Manager during processing to take care of system security restrictions. This feature was tested with code that was sent in the right format but did not actually respond to the challenge question but attempted to perform other functions that are illegal and prohibited on the system.

Sending simply java application code that tried to perform illegal operations on execution tested the Java security features. As expected, the security manager trapped the execution and the code failed to execute successfully. The contestant is notified that the entry did not execute properly and there is no harm done to the system. The details in table 5.4 give the results of one such testing session.

Entry Code	Compile Status	Execution Status	Output Match
Class contest, Valid code	Yes	Yes	Yes
Class contest, Read from a file	Yes	No	-
Class contest, Write to a new file	Yes	No	-
Class contest, Listen to permitted port	Yes	Yes	No
Class contest, Illegal socket connection	Yes	No	-

Table 6.4: Security Test Results

6.4 Web List Testing

The processing, and evaluation of entries was tested with a wide range of combinations to ensure correctness. Now, it needed to be tested that the web result listing also performed in correspondence with the rest of the system. The contents of data files created and used by the web report scripts were examined.

The successful qualified entries were sent in random order to ensure that the listing always sorted and stored the data properly. It was ascertained that the winning entry was indeed the one that had the smallest class file size. While this testing was in progress, a bug was discovered in the software. The web listing data files was storing the size of the source code and also of the corresponding class file. It was found that the sorting was being done on the source code size field but display on web page was based on the class file size. Thus, an entry that had a smaller source code but larger class file turned out to be the winning one. This bug was fixed and the system underwent another round of extensive testing.

All care has been taken to ensure consistent and reliable performance and the Java Challenge software is now ready for usage.

CHAPTER 7

DATA COLLECTION METHODOLOGY

The Java Challenge software project is a successful design and implementation of an efficient mechanism for conducting an online programming contest. As the application software is used for running a contest, it shall also aid in collection of appropriate data for software engineering analysis. The entry form designed as part of the contest system acts as the medium of data collection.

The Java Challenge Entry Form as shown in figure 7.1 collects all relevant information about the contestant's identity and computing methodology being implemented in the contest entry being submitted. The entry form initially procures data about the contestant's contact e-mail address to be used as authentication criteria, the academic status and major field of study. This data helps in identifying the knowledge base of the contestant and in estimating the extent of exposure to computing techniques.

The next section of the entry form is entirely based on the coding strategy and efforts put in by the contestant in designing the submission code. The data about the effort being single or group shall give an indication of the relative performance levels in both cases. The time spent is a measure of the coding capacity of the contestant party and it also gives insight into the planning and controls mechanisms used by the programmer.

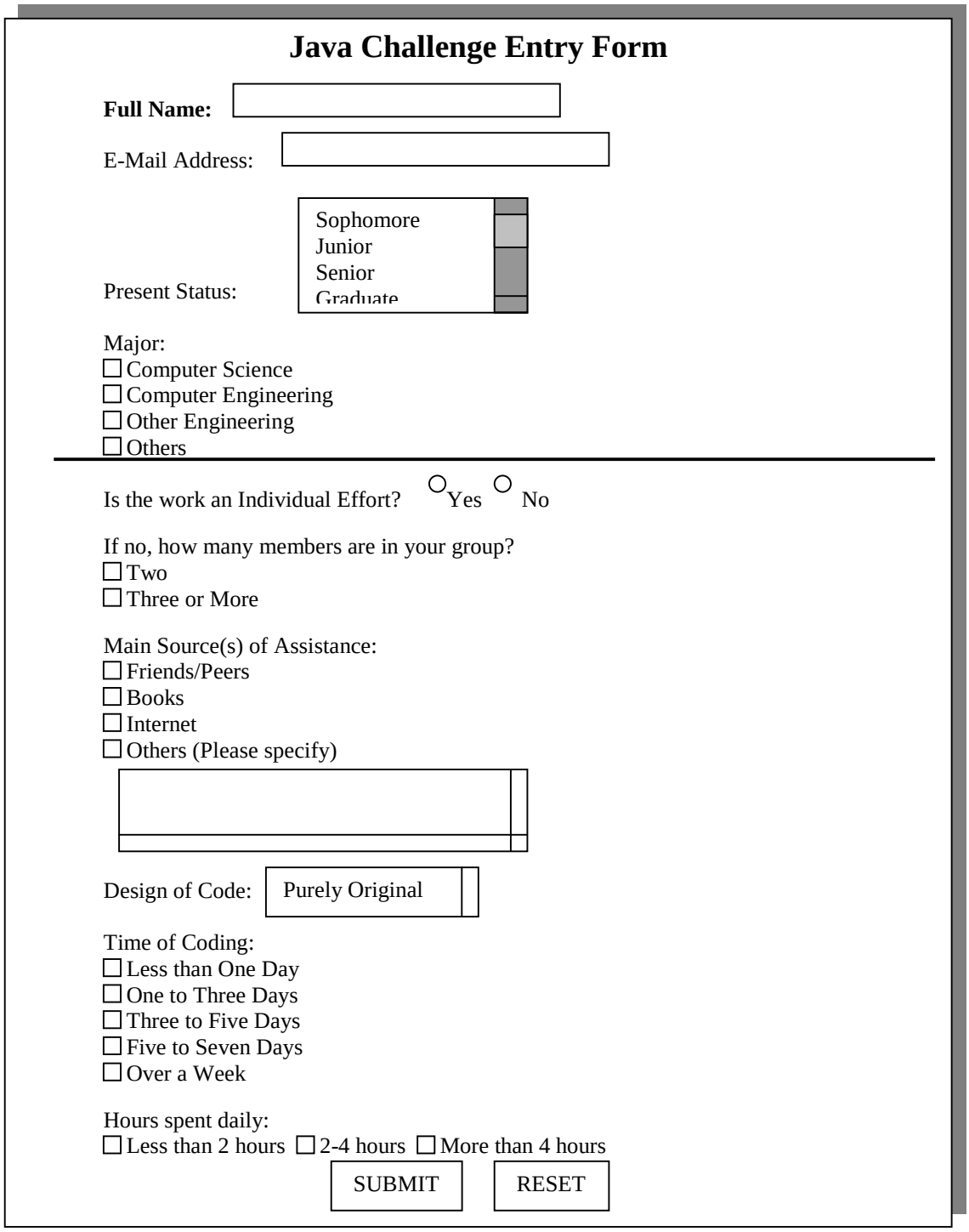

Figure 7.1: Java Challenge Entry Form

All this data is archived in the form of tables. The form contents are stored as data files that can later be analyzed for studying software-engineering features of programming in a restrained set up as a contest. Detailed studies can be undertaken to correlate various parameters for which data has been collected. Statistical analysis of this kind shall aid in predicting software development trends and specifying performance characteristics. The inferences drawn by detailed analysis of such data has great potential in proposing ways of improving software quality and monitoring development process for improved productivity and better efficiency.

The entry form is to be filled by any candidate who wishes to be qualified as the contest participant. This has been made mandatory to assist in the collection of real time data. All the work done on the data collected shall be based on the underlying assumption that the contestant has given true details and all relevant facts. Considering the fact that the form is user friendly and short, it is highly unlikely that anyone will give incorrect details. Moreover, the detailed analysis of individual trends through the entire contest is a motivation to give out correct facts as it shall finally reflect on the candidate's capability and relative performance.

When the contest is run, all data is archived in files and becomes available for detailed study and statistical analysis.

CHPATER 8

RESULTS AND CONCLUSIONS

The literature survey and research in areas related to software development and engineering showed that the task of creating complete software for conducting on-line programming contest is not an often considered and seriously undertaken task by the developing community. It brought to light various issues that need to be taken care of when designing such software package. Issues that seemed trivial were discovered to be actually very extensive and requiring deep study.

The Java Challenge Software Project was a well-planned and full fledged software project. Extensive planning was done to maintain time deadlines and the product was successfully created in reasonable time. All the phases involved in the development were undertaken with an eye for detail and tremendous technical skill was put in the efforts involved the entire process.

The JC software project has followed software-engineering principles to the maximum possible extent at all stages. The design options have been extensively researched and carefully considered for the best choice in the current situation. The system has been created as a complete, fully automated, secure and reliable tool for conducting on line programming contests. It makes use of state of the art technology and uses the latest tools for resolving various issues of system security and robustness. The software has been successfully implemented and its features extensively tested to ensure correct performance.

The Java Challenge Software Project started out as a vague vision of a collective all-inclusive package for conducting programming contests. The research work has culminated into the transformation of the vision into a reality by successfully creating the package for conducting programming contests via e-mail and making it available for ready usage.

Appropriate provisions are provided for all relevant data to be collected and archived for analysis at a later time. The coding characteristics of individual contestants are collected and the accuracy of this data is purely a matter of honesty on the part of the contestant. The task of analyzing the data is beyond the scope of this research work.

CHAPTER 9

FUTURE RECOMMENDATIONS

Future research could concentrate on extending the current package for programming languages other than Java. Full-fledged technical support in the form of manuals shall go a long way in aiding further enhancements. The software package has potential to be ported to other platforms. In view of the expanding reach of the web technology, the mode of submission can be redesigned for web based entries. This would make the contest and the package more widely usable. Even in the current mode of entry acceptance, enhancements are possible to validate messages that have the code and other details along with it as the content of the e-mail message. This requires a sophisticated means of mail filtering and file content manipulation.

The data collected during the course of running the contest is an information pool that can be appropriately tapped for software engineering research. Criteria for Selection of questions shall decide the direction of analysis and the conclusions that can be drawn from it.

REFERENCES

1. Vliet, Hans van, *Software Engineering Principles and Practice*, John Wiley& Sons, 1993.

2. Scacchi, W., "Models of Software Evolution: Life Cycle and Process", SEI-CM-10-1.0, Curriculum Module, Software Engineering Institute

3. http://members.tripod.com/~POTM

4. http://www.ioccc.org

5. http://www.cps.enel.ucalgary.ca

6. http://olympiads.win.tue.nl/ioi

7. http://www.uwp.edu/academic/mathematics/usaco.htm

8. http://www.acsl.org/

9. http://acm.baylor.org/

10. Grinzo, L., Dr Dobb's Journal, May 1999, pp 121-125

11. http://www.ecs.csus.edu/pc2/

12. http://acm.gui.uva.es/problemset/computer.html

13. Naur, NATO conference, 1968

14. IEEE standard Glossary of Software Engineering Terminology, 1983

15. Tian, J., Lu, P., Palma, J., "Test Execution Based Reliability Measurement And Modeling for Large Commercial Software", IEEE Transactions on Software Engineering, May 1995

16. Gutjhar, W. J., "Optimal Test Distributions for Software Failure Cost Estimation", IEEE Transactions on Software Engineering, March 1995

17. http://hoohoo.nsca.uiuc.edu/

18. http://www.cc.ukans.edu/~acs/docs

19. , http://pobox.com/~djb/qmail.html

20. Cormen, H. T., Leiserson, C. E. and Rivest, R. L., *Introduction to Algorithms*, McGraw Hill, 1990

21. http://java.sun.com/docs

22. Wall, L. and Schwartz, R. L., *Programming Perl*, O'Reilly & Associates, Inc.

23. McCall, J. A., Richards, P. K. and Walters, G. F., "Factors in Software Quality", RADC-TR-77-369, USDoC, 1977

24. Boehm, "Software Quality Studies", 1978

25. http://www.delorie.com/gnu/docs/screen/

26. Jaworski, J., *Java 1.2 Unleashed: The Comprehensive Solution*, Sams, 1998

27. Komarinski, M. F. and Collett, C., *Linux System Administration Handbook*, Prentice Hall, 1998

28. Sobell, M. G., *A Practical Guide to the Unix System*, Addison-Wesley, 1995

Appendix A

Evaluation Scripts

```
 ********************************************************************************
*
This is the script processing.pl that processes the entries stored in java files. It compiles, executes
and verifies the resulting output to generate log of the contest entry results.
********************************************************************************
#!/usr/bin/perl
require "ctime.pl";
### This is the script for processing entries S* files
system('cd /usr07/karuna/');
#system('pwd');
# to generate the results file
open(RESULTS, '>student_results');
open(LOG,'>>proc.log');
open(ACCESS, '>>security_test.dat');
print LOG "Entered processing.pl\n";
# for all student submissions....
foreach $file (<S*-*.java>){
  print LOG "Processing $file...\n";
  open (FILE, $file) || warn "Cannot open file!\n";
  ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size, $atime, $mtime, $ctime, $blksize,
   $blocks)=stat(FILE);
  # recording the submission time
  $submitted=$mtime;
  open (PROG, '>contest.java');
  while (<FILE>) {
    if (/^To:/){
      s/^To: //;
      s/\n//;
      $who=$_;
    } else {
      print PROG unless /^$/;
    }
  }
  close (PROG);

  # getting all properties of the file being evaluated
  open (PROG, '>>contest.java');
  ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size, $atime, $mtime, $ctime, $blksize,
   $blocks)=stat(PROG);
  close(PROG);
  print RESULTS "$size $submitted $who $file";

  # Start the evaluation: try to compile
  #print LOG "compiling...\n";
  $status=system('/projects/jcontest/Solaris_JDK_1.2.1_02/bin/javac contest.java 2> comerr.dat');
  if ($status !=0){
     system('grep -c "in a file called" comerr.dat > cerr_type.dat');
     open(ERR,'cerr_type.dat');
     while (<ERR>){
        ($errornum,$junk)=split();
     }
```

```
    if($errornum > 0){
       print RESULTS " -1 Noname Nexec Incorrect\n";
       $status=5;
       goto t1;
    }

   #print LOG "failed to compile...\n";
   print RESULTS " -1 Ncompile Nexec Incorrect\n";
   $status=1;
   goto t1;
  }
  open (PROG, '>>contest.class');
  ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size, $atime, $mtime, $ctime, $blksize,
$blocks)=stat(PROG);
  close(PROG);
  print RESULTS " $size";


  # try to execute
  #print LOG "executing...\n";
  print RESULTS " Ycompile";
  #capture output of run in output file
  $status=system('/projects/jcontest/Solaris_JDK_1.2.1_02/bin/java -Djava.security.manager -
Djava.security.policy=="/usr07/karuna/ALL/cs374/rwpolicy" contest 1> output 2> exerr.dat');
  #$status=system('java contest > output');
  if ($status !=0){
     system('grep -c AccessControlException exerr.dat > err_type.dat');
     open(ERR,'err_type.dat');
     while (<ERR>){
           ($errornum,$junk)=split();
     }

     print ACCESS "$who:$errornum\n";
     if($errornum > 0){
           print RESULTS " Illegal Incorrect\n";
           $status=3;
           goto t1;
     }

   #print LOG "failed to execute...\n";
   print RESULTS " Nexec Incorrect\n";
   $status=2;
   goto t1;
  }

  # compare to check if output is correct
  #print LOG "verifying output...\n";
  print RESULTS " Yexec";

  #this file comparison might be changed depending on contest question
```

```
   $status=system ('diff stdOutput output');
  if ($status !=0){

    #print LOG "Incorrect output...\n";
    print RESULTS " Incorrect\n";
    $status=4;
    goto t1;
  }
  #print  LOG "correct...\n";
  print RESULTS " Correct\n";
 t1:close (FILE);

  print "Logging...";
  if ($status == 0) {
    print "Correct \n";
  }
  if ($status == 1) {
    print "Nocompile ";
  }
  if ($status == 2) {
    print "Noexec ";
  }
  if ($status == 3) {
    print "Illegal ";
  }
  if ($status == 4) {
    print "Incorrect ";
  }
  if ($status == 5) {
      print "Noname ";
  }
}
close (RESULTS);
close (ACCESS);
close(ERR);

# for all non-student submissions....

system ('sort -n < student_results > sorted_stu_results');
#print LOG "Done with processing\n";
$date=&ctime(time);
print LOG "$date";
#($sec,$min,$hr,$mday,$mon,$yr,$wday,$yday,$isdst)=localtime(time);
#print LOG "$hr:$min:$sec;$mday,$mon,$yr:$wday,$yday,$isdst\n";
print LOG "===*===\n";
close LOG;

*****************************************************************************
```

```
 ********************************************************************************
*
This is the script mailing.pl that sends e-mail notifications to contestants about the status of their
entry. It determines the result by looking at the contents of data files created by processing.pl
********************************************************************************
#!/usr/bin/perl
require "ctime.pl";
### This is the script for sending mails to the contestants about their
###submission results
system('cd /usr07/karuna/');
system('cat sorted_stu_results >> cum_stu_results');
system('sort -n < cum_stu_results | uniq > sum_stu_results');
open (FILE, 'sorted_stu_results');
open(LOG,'>>mail.log');
# sending message to everyone on the list...
print LOG "Entered mailing.pl\n";
while (<FILE>){
  #print LOG "Entered while of mailing.pl\n";
  open (MAIL, '>mail_Mesg');
  ($size, $time, $who, $file, $csize, $cs, $es, $rs)=split(' ');
  $realtime=&ctime($time);
  $realtime=~ s/\n//;
  print MAIL "To: $who\n";
  print MAIL "From: karuna\@csee.wvu.edu\n";
  print MAIL "Cc: karuna\@csee.wvu.edu\n";

  if ($cs=~ /Noname/){
     print MAIL "Subject: Did not compile: class name error!\n\n";
     print MAIL "Dear $who, The following program:\n\n";
     open(PROG, $file);
     while(<PROG>) {
     if(/^To: /)
     { }
     else
     {print MAIL;}
  }
     close (PROG);
    #print LOG "did not compile mail sent out\n";
     print MAIL "\n\n did not compile due to incorrect class
name...Please try again.";
     goto t1;
  }

  #submission did not compile...
  if ($cs=~ /Ncompile/){
    print MAIL "Subject: Did not compile!\n\n";
    print MAIL "Dear $who, The following program:\n\n";
    open(PROG, $file);
    while(<PROG>) {
      if(/^To: /)
            { }
```

```
      else
          {print MAIL;}
   }
   close (PROG);
   #print LOG "did not compile mail sent out\n";
   print MAIL "\n\n did not compile...Please try again.";
   goto t1;
  }

  #submission has security exception...
  if ($es=~ /Illegal/){
     print MAIL "Subject: Illegal operation performed!\n\n";
     print MAIL "Dear $who, The following program:\n\n";
     open(PROG, $file);
     while(<PROG>) {
     if(/^To: /)
     { }
     else
     {print MAIL;}
  }
     close (PROG);
     print MAIL "\n\n performed illegal operation...Please see the web page for rules and try
again.";
     goto t1;
  }
  #submission did not execute...
  if ($es=~ /Nexec/){
    print MAIL "Subject: Did not execute!\n\n";
    print MAIL "Dear $who, The following program:\n\n";
    open(PROG, $file);
    while(<PROG>) {
     if(/^To: /)
          { }
     else
          {print MAIL;}
    }
    close (PROG);
    #print LOG "did not execute mail sent out\n";

    print MAIL "\n\n did not execute properly...Please try again.";
    goto t1;
  }

  #submission did not give correct output...
  if ($rs=~ /Incorrect/){
    print MAIL "Subject: Incorrect output!\n\n";
    print MAIL "Dear $who, The following program:\n\n";
    open(PROG, $file);
    while(<PROG>) {
     if(/^To: /)
          { }
```

```
      else
           {print MAIL;}
   }
   close (PROG);
   #print LOG "did not give right o/p mail sent out\n";
   open(ERR,'security.dat');
   while (<ERR>){
     ($sender,$number)=split(':');
     if(($who eq $sender) && ($number>0)){
           print MAIL "\n performed an illegal operation.";
     }
   }
   close (ERR);
   print MAIL "\n\n did not produce correct output...Please try again.";
   goto t1;
 }

 # submission was all fine...
 print MAIL "Subject: Correct!!!\n\n";
 print MAIL "Dear $who, The following program:\n\n";
 open(PROG, $file); while(<PROG>) {
   if (/^To: /)
     { }
   else
     {print MAIL;}
 }
 close (PROG);
 #print LOG "all correct mail sent out\n";
 print MAIL "\n\n is correct and has qualified! Its class size is
          $csize and was received on $realtime.";
 goto t1;

t1: print MAIL "\nSee http://www.csee.wvu.edu/~karuna/contest.html for current standings!\n";
 close (MAIL);
 system('/usr/lib/sendmail -t < mail_Mesg');
}
close(FILE);

system('rm sorted_stu_results');
#print LOG "Done with sending mails to contestants\n";
$date=&ctime(time);
print LOG "$date";
#($sec,$min,$hr,$mday,$mon,$yr,$wday,$yday,$isdst)=localtime(time);
#print LOG "$hr:$min:$sec;$mday,$mon,$yr:$wday,$yday,$isdst\n";
print LOG "===*===\n";
close LOG;

*******************************************************************************
```

```
 ********************************************************************************
*
This is the script webrpt.pl that updates the result listing on the web by determining the order of
winning entries and looking at data file generated by the jcForm.pl to see if the contestant has
filled the entry form and is eligible.
********************************************************************************
#!/usr/bin/perl
require "ctime.pl";
### This script updates the web page of current standing in the challenge
system('cd /usr07/karuna/');
system('cp /usr07/karuna/public_html/cgi-bin/names.dat /usr07/karuna/');
system('mv /usr07/karuna/S*-*.java /usr07/karuna/entries/');
open(WEB,'>result.html');
open(STU,'>stu_file');
open(LOG,'>>webrpt.log');
print LOG "Entered web_rpt.pl\n";
print WEB <<WEB_PAGE_HEADER;
<HTML>
<HEAD>
<TITLE>Student Scores</TITLE>
</HEAD>
<BODY BGCOLOR="brown" TEXT="white">
<CENTER><H2>Current Standings for Java Challenge</H2><HR></CENTER>
WEB_PAGE_HEADER

$cnt=1;
open (RESULTS, 'sum_stu_results');
#print LOG "opened ssr\t";
while (<RESULTS>){
  #print LOG "while1...\t";
  ($size, $time, $who, $file, $csize, $cs, $es, $rs)=split(/ /);
  $realtime=&ctime($time);
  $realtime=~ s/\n//;
  # separate login name from the $who value
  ($logname, $domname)=split(/@/,$who);
  #print "Separated logname:$logname\n";
  open(DATA, 'names.dat');
  while (<DATA>){
    #print LOG "while DATA\t";
    ($formname,$formmail)=split(/:/);
    ($flogname,$fdomname)=split(/@/,$formmail);
    if($logname eq $flogname){
      $grepresult=0;
      #print "$grepresult";
      goto l1;
    }
    else{
      $grepresult=1;
    }
  }
  #print LOG "b4 l1\t";
```

```
  l1:close DATA;
  #print "Separated flogname too...\n";
  #print "\n$flogname ::: $logname";

  if($cs=~ /Ycompile/ && $es=~ /Yexec/ && $rs=~ /Correct/){
    #print LOG "if all fine\n";
    if($cnt==1){
      #print LOG "if cnt=1 part\t";
      print STU "\n Java Challenge Winner is $who!!!\n";
      print WEB<<STR_BLOCK;
<CENTER><H2><I>Current Leader is <BLINK>$who</BLINK>!!!</I></H2></CENTER>
<TABLE BORDER=1>
<TH> Rank<TH>Size (in bytes)<TH>Time Received<TH>Person<TH>Entry Form</TH>
STR_BLOCK
    }
    if ($grepresult==0){
      #print LOG "if grepr=0\t";
      print WEB
"<TR><TD>$cnt</TD><TD>$csize</TD><TD>$realtime</TD><TD>$who</TD><TD>Yes</
TD>\n"
    }
    else{
      #print LOG "else grepr\t";
      print WEB
"<TR><TD>$cnt</TD><TD>$csize</TD><TD>$realtime</TD><TD>$who</TD><TD>No</T
D>\n"
    }
    print STU "\n$cnt $csize $realtime $who $grepresult\n";
    $cnt ++;
    #print LOG "cnt++\t";
  }
}
close (RESULTS);
print WEB<<WEB_PAGE_FOOTER;
</TABLE><BR><BR><BR>
Contest details available at <a
href="http://www.csee.wvu.edu/~karuna/contest.html">http://www.csee.wvu.edu/~karuna/contes
t.html</a>\n
</BODY>
</HTML>
WEB_PAGE_FOOTER

close WEB;
close STU;
system('cp /usr07/karuna/result.html /usr07/karuna/public_html/');
system('chmod ugo+rx /usr07/karuna/public_html/result.html');
#system('rm -f /usr07/karuna/result.html');
#system('cat stu >> stu_data');
#print LOG "\nWeb page updated...\n";
$date=&ctime(time);
#($sec,$min,$hr,$mday,$mon,$yr,$wday,$yday,$isdst)=localtime(time);
```

```
  print LOG "$date";
#print LOG "$hr:$min:$sec;$mday,$mon,$yr:$wday,$yday,$isdst\n";
print LOG "===***===\n";
close LOG;
```

****

```
 ********************************************************************************
*
This is the script jcForm.pl that processes the data filled in the entry form on the web. It archives
the data, sends a response html page and also sends e-mail confirmation of contestant eligibility.
********************************************************************************
#!/usr/bin/perl
#capture the data from the form:
require "cgi-lib.pl";
&ReadParse;

#send html document MIME type:
print "Content-Type:text/html\n\n";

#the acceptance web page:
print<<END_OF_MESSAGE;
<html>
<title>Acceptance Page</title>
<body><h1>Thank you for Contesting!</h1>
Thank you $in{'name'}, for filling the form!<br>
You are now eligible to contest in the Java challenge.<br>
Remember to send the contest entry from <b>$in{'email'} </b><BR>
with the subject as <B>(CONTEST)</B> for successful acceptance.<BR>
Please note that entry evaluation and web posting updates happen every quarter hour.
<BR><BR><BR>
<center>Visit <a href="http://www.csee.wvu.edu/~karuna/contest.html">
http://www.csee.wvu.edu/~karuna/
contest.html</a> for updates!!!</center>
</body></html>
END_OF_MESSAGE

#send an e-mail to the contestant:
open (MAIL, "| /usr/lib/sendmail -oi -t" );
print MAIL <<MESSAGE_TO_CONTESTANT;
To:$in{'email'}
From: karuna\@www.csee.wvu.edu
Cc: karuna\@www.csee.wvu.edu
Subject: Java Challenge Entry Form

Dear $in{'name'}:
Your entry form has been received and recorded. No matter how many entries
you send in for one contest problem, you have to fill the entry form only
ONCE! Now that you have done that, you DO NOT have to repeat that task.
Please note that the form shall have be filled again if you have make major
changes to your approach or strategy.
Remember to send your solution by $in{'email'} e-mail address!
Thanks and Happy Coding!!!

http://www.csee.wvu.edu/~karuna/contest.html
MESSAGE_TO_CONTESTANT
close MAIL;
```

```
 #send a message to the master that a new entry has come n been replied
#to via e-mail:
open (MAIL, "| /usr/lib/sendmail -oi -t");
print MAIL <<MESSAGE_TO_MASTER;
To:karuna\@csee.wvu.edu
From: karuna\@csee.wvu.edu
Subject: Another J_C Entry Form

Another entry is reported from the entry-form web-page.
e-mail has been sent to $in{'name'} at $in{'email'} as confirmation.
MESSAGE_TO_MASTER
close MAIL;

#record contents of the form in a text file:
@sources=split("\0", $in{'source'});
if ($in{'team'} eq "yes"){
open (DATA_FILE,">> data.txt");
print DATA_FILE<<DATA_RECORD;
$in{'name'}\t$in{'status'}\t$in{'major'}\tAlone\t$in{'subNum'}\t@sources\t$in{'design'}\t$in{'da
ys'}\t$in{'hours'}\n$in{'othS'}\n
DATA_RECORD
close DATA_FILE;}
else{
open(DATA_FILE,">> data.txt");
print DATA_FILE<<DATA_RECORD;
$in{'name'}\t$in{'status'}\t$in{'major'}\t$in{'group'}\t$in{'subNum'}\t@sources\t$in{'design'}\t$
in{'days'}\t$in{'hours'}\n$in{'othS'}\n
DATA_RECORD
close DATE_FILE;}

#Keep a log of contestants
open(NAME_FILE, ">> names.dat");
print NAME_FILE<<NAME_DATA;
$in{'name'}:$in{'email'}\n
NAME_DATA
close NAME_FILE;

*********************************************************************************
```

```
 ********************************************************************************
*
This is the master.shscr script, the shell script that executes all the perl scripts in the right order
and also generates corresponding log files.
********************************************************************************
#!/usr/bin/tcsh

cd /usr07/karuna/
echo "ps" >> processing.log
perl /usr07/karuna/processing.pl 2> err_proc.log
cat err_proc.log >> processing.log
echo "ms" >> mailing.log
perl /usr07/karuna/mailing.pl 2> err_mail.log
cat err_mail.log >> mailing.log
echo "ws" >> webrpt.log
perl /usr07/karuna/webrpt.pl 2> err_web.log
cat err_web.log >> webrpt.log

********************************************************************************
```

Appendix B

Mail Filter and Automation Scripts

```
 ****************************************************************************
*
This is the .forward file that triggers the procmail filter every time a new message arrives in the mailbox.
****************************************************************************

"|IFS=' ' && p=/usr/local/bin/procmail && test -f $p && exec $p -Yf- || exit 75 #karuna"


****************************************************************************
This is the customized procmail filter for processing e-mail messages that are received as contest entries.
****************************************************************************
# check if all paths are reachable
#TMPDIR=./tmp
PATH=/bin:/usr/bin:/usr/local/bin:/usr/ucb:/bin:.
MAILDIR=$HOME/Mail
DEFAULT=/var/spool/mail/karuna
LOGFILE=$HOME/.maillog
#LOCKFILE=$HOME/lockmail

:0
* ^Subject:.*\(CONTEST\)
{
        :0c
        | (formail -brt -I 'Subject: Re: Your Java Challenge Submission' \
        -I 'From: Karuna Annavajjala <karuna@csee.wvu.edu>' ;\
        cat $HOME/.goodmesg) | $SENDMAIL -t

        :0c
        | (formail -brt -I 'Subject: Re: A "S" Java Challenge Submission' \
        -I 'From: Karuna Annavajjala <karuna@csee.wvu.edu>' \
        -I 'To:Karuna Annavajjala <karuna@csee.wvu.edu>' ; \
        cat $HOME/.mymesg) | $SENDMAIL -t

        :0c
        | formail -k -brt -X To: >/usr07/karuna/S$$-`date +%m%d%y%H%M%S`.java

        :A
        contest
}

:0
* ^Subject:.*\(NOCONTEST\)
{
        :0c
        | (formail -brt -I 'Subject: Re: Your Java Challenge Submission' \
        -I 'From: Karuna Annavajjala <karuna@csee.wvu.edu>' ;\
        cat $HOME/.goodmesg) | $SENDMAIL -t

        :0c
```

```
          | (formail -brt -I 'Subject: Re:A "X" Java Challenge Submission' \
          -I 'From: Karuna Annavajjala <karuna@csee.wvu.edu>' \
          -I 'To:Karuna Annavajjala <karuna@csee.wvu.edu>' ;\
          cat $HOME/.mymesg) | $SENDMAIL -t

          :0c
          | formail -k -brt -X To: >/usr07/karuna/X$$-`date +%m%d%y%H%M%S`.java

          :A
          contest
}

:0
* ^Subject:.*CONTEST
#* ^From:.*.wvu.edu
{
          :0c
          | (formail -brt -I 'Subject: Re: Your Java Challenge Submission' \
          -I 'From: Karuna Annavajjala < karuna@csee.wvu.edu>' ; \
          cat $HOME/.badmesg) | $SENDMAIL -t
}

# anything that has not been handled here will go to $DEFUALT using the LOCKFILE
```

```
 ********************************************************************************
*
```
This is the testbac shell script that acts as the daemon script on backus system to execute the master shell script at regular time intervals.
```
********************************************************************************
#!/usr/bin/tcsh

while (1 == 1)
master.shscr 2> err_testbac.log
date>>testbac.log
echo "***">>testbac.log
sleep 1800
end
```

```
********************************************************************************
```
This is the syntax for the crontab file to place the script execution on the cron job list.
```
********************************************************************************
1,15,30,45 * * * * /usr07/karuna/master.shscr
```

```
********************************************************************************
```
This is a segment of procmail filter to demonstrate its usage for automation purposes. The script execution can be triggered from the action line of the filter message.
```
********************************************************************************
# check if all paths are reachable
#TMPDIR=./tmp
PATH=/bin:/usr/bin:/usr/local/bin:/usr/ucb:/bin:.
MAILDIR=$HOME/Mail
DEFAULT=/var/spool/mail/karuna
LOGFILE=$HOME/.maillog
#LOCKFILE=$HOME/lockmail

:0
* ^Subject:.*\(CONTEST\)
{
        :0c
        | formail -k -brt -X To: >/usr07/karuna/S$$-`date +%m%d%y%H%M%S`.java

        :0c
        |/usr07/karuna/master.shscr

        :A
        contest
}
```

Appendix C

jdk 1.2 Security Manager

(http://java.sun.com/)

Examples of security policy files:

```
grant signedBy "signer_names", codeBase "URL" {
    permission permission_class_name "target_name", "action",
        signedBy "signer_names";
    ....
    permission permission_class_name "target_name", "action",
        signedBy "signer_names";
  };
```

```
grant codeBase "file:/C:/somepath/api/" { ... }
```

```
grant signedBy "sysadmin", codeBase "file:/home/sysadmin/*" {
permission java.security.SecurityPermission "Security.insertProvider.*";
permission java.security.SecurityPermission "Security.removeProvider.*";
permission java.security.SecurityPermission "Security.setProperty.*"; };
```

The security policy file created for Java Challenge Software:

```
/* AUTOMATICALLY GENERATED ON Tue Jul 06 19:26:10 EDT 1999*/
/* DO NOT EDIT */

grant codeBase "file:/usr07/karuna/-" {
  permission java.io.FilePermission "/var/temp/-", "read";
  permission java.net.NetPermission "requestPasswordAuthentication";
  permission java.net.SocketPermission "*:65000", "resolve, listen";
};
```

The Permission Classes

The permission classes represent access to system resources. The java.security.Permission class is an abstract class and is subclassed, as appropriate, to represent specific accesses.

As an example of a permission, the following code can be used to produce a permission to read the file named "abc" in the /tmp directory:

```
perm = new java.io.FilePermission("/tmp/abc", "read");
```

New permissions are subclassed either from the Permission class or one of its subclasses, such as java.security.BasicPermission. Subclassed permissions (other than BasicPermission) generally belong to their own packages. Thus, FilePermission is found in the java.io package.

A crucial abstract method that needs to be implemented for each new class of permission is the `implies` method. Basically, "a implies b" means that if one is granted permission "a", one is naturally granted permission "b". This is important when making access control decisions.

Associated with the abstract class java.security.Permission are the abstract class named java.security.PermissionCollection and the final class java.security.Permissions.

Class java.security.PermissionCollection represents a collection (i.e., a set that allows duplicates) of Permission objects for a single category (such as file permissions), for ease of grouping. In cases where permissions can be added to the PermissionCollection object in any order, such as for file permissions, it is crucial that the PermissionCollection object ensure that the correct semantics are followed when the `implies` function is called.

Class java.security.Permissions represents a collection of collections of Permission objects, or in other words, a super collection of heterogeneous permissions.

Applications are free to add new categories of permissions that the system supports. How to add such application-specific permissions is discussed later in this document.

Now we describe the syntax and semantics of all built-in permissions.

C.1.1 java.security.Permission

This abstract class is the ancestor of all permissions. It defines the essential functionalities required for all permissions.

Each permission instance is typically generated by passing one or more string parameters to the constructor. In a common case with two parameters, the first parameter is usually "the name of the target" (such as the name of a file for which

the permission is aimed), and the second parameter is the action (such as "read" action on a file). Generally, a set of actions can be specified together as a comma-separated composite string.

C.1.2 java.security.PermissionCollection

This class holds a homogeneous collection of permissions. In other words, each instance of the class holds only permissions of the same type.

C.1.3 java.security.Permissions

This class is designed to hold a heterogeneous collection of permissions. Basically, it is a collection of java.security.PermissionCollection objects.

C.1.4 java.security.UnresolvedPermission

Recall that the internal state of a security policy is normally expressed by the permission objects that are associated with each code source. Given the dynamic nature of Java technology, however, it is possible that when the policy is initialized the actual code that implements a particular permission class has not yet been loaded and defined in the Java application environment. For example, a referenced permission class may be in a JAR file that will later be loaded.

The UnresolvedPermission class is used to hold such "unresolved" permissions. Similarly, the class java.security.UnresolvedPermissionCollection stores a collection of UnresolvedPermission permissions.

During access control checking on a permission of a type that was previously unresolved, but whose class has since been loaded, the unresolved permission is "resolved" and the appropriate access control decision is made. That is, a new object of the appropriate class type is instantiated, if possible, based on the information in the UnresolvedPermission. This new object replaces the UnresolvedPermission, which is removed.

If the permission is still unresolvable at this time, the permission is considered invalid, as if it is never granted in a security policy.

C.1.5 java.io.FilePermission

The targets for this class can be specified in the following ways, where directory and file names are strings that cannot contain white spaces.

```
file
directory (same as directory/)
directory/file
directory/* (all files in this directory)
* (all files in the current directory)
directory/- (all files in the file system under this directory)
- (all files in the file system under the current directory)
"<<ALL FILES>>" (all files in the file system)
```

Note that "<<ALL FILES>>" is a special string denoting all files in the system.

The actions are: **read**, **write**, **delete**, and **execute**. Therefore, the following are valid code samples for creating file permissions:

```
import java.io.FilePermission;

FilePermission p = new FilePermission("myfile", "read,write");
FilePermission p = new FilePermission("/home/gong/", "read");
FilePermission p = new FilePermission("/tmp/mytmp",
"read,delete");
FilePermission p = new FilePermission("/bin/*", "execute");
FilePermission p = new FilePermission("*", "read");
FilePermission p = new FilePermission("/-", "read,execute");
FilePermission p = new FilePermission("-", "read,execute");
FilePermission p = new FilePermission("<<ALL FILES>>", "read");
```

The `implies` method in this class correctly interprets the file system. For example, FilePermission("/-", "read,execute") implies FilePermission("/home/gong/public_html/index.html", "read"), and FilePermission("bin/*", "execute") implies FilePermission("bin/emacs19.31", "execute").

```
Note: Most of these strings are given in platform-dependent
format.
```

It is necessary that the strings be given in platform-dependent format until there is a universal file description language. Note also that the use of meta symbols such as "*" and "-" prevents the use of specific file names. Also note that a target name that specifies just a directory, with a "read" action, as in

```
FilePermission p = new FilePermission("/home/gong/", "read");
```

means you are only giving permission to list the files in that directory, not read any of them. To allow read access to files, you must specify either an explicit file name, or an "*" or "-", as in

```
FilePermission p = new FilePermission("/home/gong/myfile",
"read");
FilePermission p = new FilePermission("/home/gong/*", "read");
FilePermission p = new FilePermission("/home/gong/-", "read");
```

And finally, note that code always automatically has permission to read files from its same (URL) location, and subdirectories of that location; it does not need explicit permission to do so.

C.1.6 java.net.SocketPermission

This class represents access to a network via sockets. The target for this class can be given as "hostname:port_range", where hostname can be given in the following ways:

```
hostname (a single host)
IP address (a single host)
localhost (the local machine)
"" (equivalent to "localhost")
hostname.domain (a single host within the domain)
hostname.subdomain.domain
*.domain (all hosts in the domain)
```

```
*.subdomain.domain
* (all hosts)
```

That is, the host is expressed as a DNS name, as a numerical IP address, as "localhost" (for the local machine) or as "" (which is equivalent to specifying "localhost").

The wildcard "*" may be included once in a DNS name host specification. If it is included, it must be in the leftmost position, as in "*.sun.com".

The port_range can be given as follows:

```
N (a single port)
N- (all ports numbered N and above)
-N (all ports numbered N and below)
N1-N2 (all ports between N1 and N2, inclusive)
```

Here N, N1, and N2 are non-negative integers ranging from 0 to 65535 (2^16-1).

The actions on sockets are **accept**, **connect**, **listen**, and **resolve** (which is basically DNS lookup). Note that implicitly, the action "resolve" is implied by "accept", "connect", and "listen" -- i.e., those who can listen or accept incoming connections from or initiate out-going connections to a host should be able to look up the name of the remote host.

Below are some examples of socket permissions.

```
import java.net.SocketPermission;

SocketPermission p = new
SocketPermission("java.sun.com","accept");
p = new SocketPermission("204.160.241.99","accept");
p = new SocketPermission("*.com","connect");
p = new SocketPermission("*.sun.com:80","accept");
p = new SocketPermission("*.sun.com:-1023","accept");
p = new SocketPermission("*.sun.com:1024-","connect");
p = new SocketPermission("java.sun.com:8000-9000",
         "connect,accept");
p = new SocketPermission("localhost:1024-",
          "accept,connect,listen");
```

Note that SocketPermission("java.sun.com:80,8080","accept") and SocketPermission("java.sun.com,javasun.sun.com","accept") are not valid socket permissions.

Moreover, because **listen** is an action that applies only to ports on the local host, whereas **accept** is an action that applies to ports on both the local and remote host, both actions are necessary.

C.1.7 java.security.BasicPermission

The BasicPermission class extends the Permission class. It can be used as the base class for permissions that want to follow the same naming convention as BasicPermission (see below).

The name for a BasicPermission is the name of the given permission (for example, "exitVM", "setFactory", "queuePrintJob", etc). The naming convention follows the hierarchical property naming convention. An asterisk may appear at the end of the name, following a ".", or by itself, to signify a wildcard match. For example: "java.*" or "*" is valid, "*java" or "a*b" is not valid.

The action string (inherited from Permission) is unused.Thus, BasicPermission is commonly used as the base class for "named" permissions (ones that contain a name but no actions list; you either have the named permission or you don't.) Subclasses may implement actions on top of BasicPermission, if desired.

Some of the BasicPermission subclasses are java.lang.RuntimePermission, java.security.SecurityPermission, java.util.PropertyPermission, and java.net.NetPermission.

C.1.8 java.util.PropertyPermission

The targets for this class are basically the names of Java properties as set in various property files. Examples are the "java.home" and "os.name" properties. Targets can be specified as "*" (any property), "a.*" (any property whose name has a prefix "a."), "a.b.*", and so on. Note that the wildcard can occur only once and can only be at the rightmost position.

This is one of the BasicPermission subclasses that implements actions on top of BasicPermission. The actions are read and write. Their meaning is defined as follows: "read" permission allows the `getProperty` method in java.lang.System to be called to get the property value, and "write" permission allows the `setProperty` method to be called to set the property value.

C.1.9 java.lang.RuntimePermission

The target for a RuntimePermission can be represented by any string, and there is no action associated with the targets. For example, RuntimePermission("exitVM") denotes the permission to exit the Java Virtual Machine.

The target names are:

```
createClassLoader
getClassLoader
setContextClassLoader
setSecurityManager
createSecurityManager
exitVM
setFactory
setIO
modifyThread
stopThread
modifyThreadGroup
getProtectionDomain
readFileDescriptor
writeFileDescriptor
loadLibrary.{library name}
```

```
accessClassInPackage.{package name}
defineClassInPackage.{package name}
  accessDeclaredMembers.{class name}
  queuePrintJob
```

C.1.10 java.awt.AWTPermission

This is in the same spirit as the RuntimePermission; it's a permission without actions. The targets for this class are:

```
accessClipboard
accessEventQueue
listenToAllAWTEvents
showWindowWithoutWarningBanner
```

C.1.11 java.net.NetPermission

This class contains the following targets and no actions:

```
requestPasswordAuthentication
setDefaultAuthenticator
specifyStreamHandler
```

C.1.12 java.lang.reflect.ReflectPermission

This is the Permission class for reflective operations. A ReflectPermission is a named permission (like RuntimePermission) and has no actions. The only name currently defined is

```
suppressAccessChecks
```

which allows suppressing the standard Java programming language access checks -- for public, default (package) access, protected, and private members -- performed by reflected objects at their point of use.

C.1.13 java.io.SerializablePermission

This class contains the following targets and no actions:

```
enableSubclassImplementation
enableSubstitution
```

C.1.14 java.security.SecurityPermission

SecurityPermissions control access to security-related objects, such as Security, Policy, Provider, Signer, and Identity objects. This class contains the following targets and no actions:

```
getPolicy
setPolicy
getProperty.{key}
setProperty.{key}
insertProvider.{provider name}
removeProvider.{provider name}
setSystemScope
setIdentityPublicKey
setIdentityInfo
printIdentity
addIdentityCertificate
```

```
removeIdentityCertificate
clearProviderProperties.{provider name}
  putProviderProperty.{provider name}
  removeProviderProperty.{provider name}
  getSignerPrivateKey
  setSignerKeyPair
```

C.1.15 java.security.AllPermission

This permission implies all permissions. Note that AllPermission also implies new permissions that are defined in the future.

Clearly much caution is necessary when considering granting this permission. These features of the jdk1.2 security model can be implemented by appropriate use of available tools of development.

KARUNA ANNAVAJJALA

E-mail: **a_karuna@hotmail.com**

EDUCATION

Master of Science in Computer Science
West Virginia University, WV. (August 1999)
GPA: 3.8/4.0

Bachelor of Science in Civil Engineering
Osmania University, India. (July 1997)
GPA: 3.8/4.0

Diploma in RDBMS
Frontier Institute of Information Technology, India (April 1997)

COMPUTER SKILLS

Operating Systems	MS DOS, UNIX, Windows 95/98, Windows NT, MAC OS, Sun Solaris, Linux (Redhat 5.x), Irix (Silicon Graphics), VAX/VMS
Platforms	IBM/PC Compatibles, IBM/DEC Workstations, MACHINTOSH
Languages	C, C++, Java, ADA, FORTRAN, PASCAL, BASIC, VB Script, HTML, PL/SQL, Perl, Unix Shell Scripting, Scheme
RDBMS	ORACLE 7.x

WORK EXPERIENCE

Teaching Assistant in the department of Computer Science
West Virginia University, Morgantown (June 1998 to June 1999)

Teaching Assistant for Organic Chemistry Laboratory.
West Virginia University, Morgantown (Aug 1997 to May 1998)

Further details and references available upon request